# File Group Administration Guide

Axiom

Version 2021.1

Syntellis
10 S. Wacker Dr., Suite 3375
Chicago, Illinois 60606
847.441.0022
www.syntellis.com

# Table of Contents

# Introduction

In Axiom, *file groups* are sets of files that are used to develop planning or financial data. File groups define the building blocks of your plan, such as templates, drivers, and calc methods, and also contain the individual plan files used to model and calculate the plan data.

This guide discusses how to create and administer file groups and their associated components.

## Intended audience

This guide is intended for administrators and other power users who are responsible for managing file groups.

## What is covered in this guide?

This guide covers the following aspects of file group administration:

- File group setup and administration
- Template administration
- Driver file administration
- Calc method administration
- Plan file utilities (Create Plan Files, Process Plan Files)

## What is not covered in this guide?

The following related topics are not covered in this guide:

- End user access and use of plan files. For information on how end users work with built plan files and their associated features, see the *Desktop Client User Guide*.
- Setup of Axiom files, including use of Axiom queries, Axiom functions, and Control Sheet settings. For more information, see the *Axiom File Setup Guide*.
- Creating form-enabled plan files and other file group components. For more information, see the *Axiom Forms and Dashboards Guide*.
- Setting up and using a plan file process to progress plan files through a defined set of edit and approval steps. For more information, see the *Plan File Process Guide*.

All documentation for Axiom can also be accessed using the Axiom Help Files.

## Axiom Client versions

This guide discusses functionality that is available in the Axiom Desktop Client (Excel Client and Windows Client). Screenshots of features may show either the Excel Client or the Windows Client. The Axiom functionality is virtually identical in both environments.

# File Group Administration

This section contains conceptual information about file groups, and explains how to create, edit, and delete file groups. Once a file group has been created, you can create and work with the individual components, such as templates, drivers, and plan files.

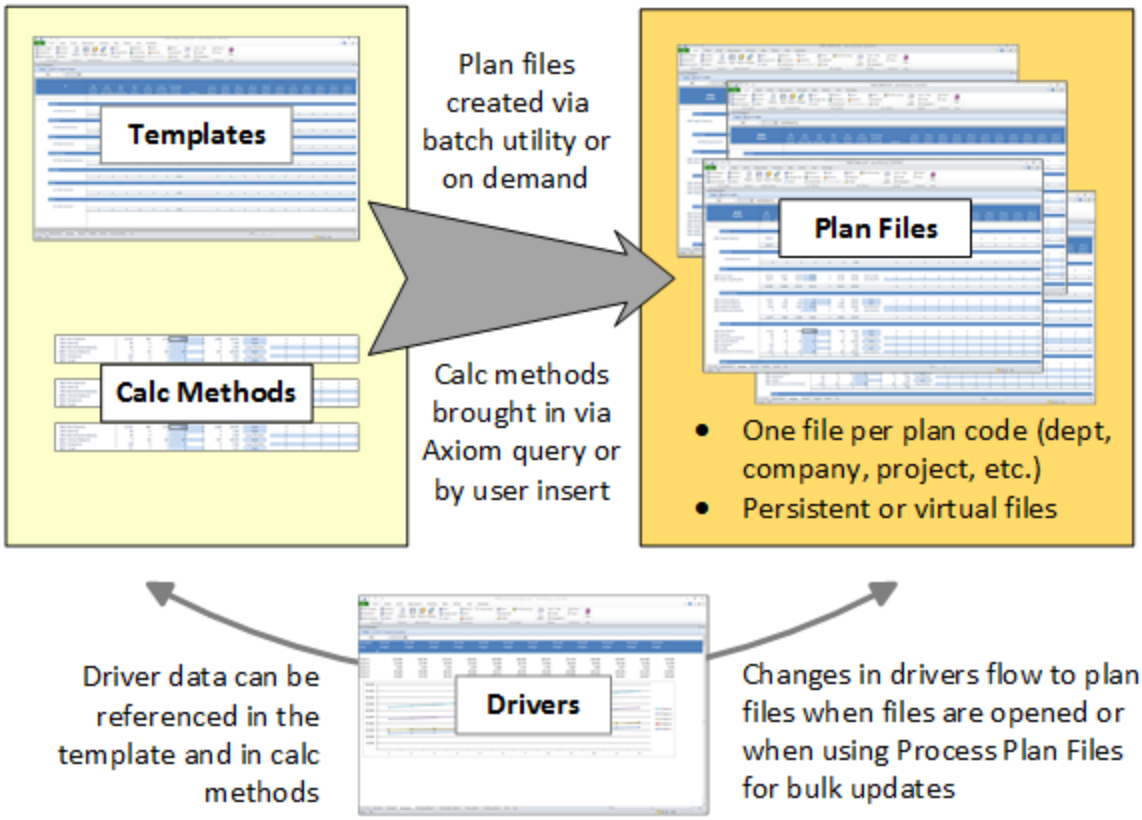
## About file groups

*File groups* are sets of files that are used to develop planning or financial data. Each installation of Axiom can have as many active file groups as desired.

### File group components

Each file group in Axiom uses the following components:

- *Templates* define the structure and user interface of the plan. Templates also define where data is brought into each sheet, and other controls.
- *Calc methods* define the calculation logic and formatting to be applied to the data in the plan. Each file group has multiple calc method libraries that hold the standard calculations for the plan.
- *Drivers* store information and statistics for the plan. Driver files control plan-wide settings, such as the current planning period, and contain plan "drivers" (such as Payroll-related rates and percentages, production or utilization statistics, and conversion rates) that are used in the plan files to calculate and spread data.
- *Plan files* are automatically created using a combination of templates, calc methods, and driver data. Once the "base case" plan is created, department managers and other users can access their plan files to complete plan inputs and adjust planning data and calculations as needed (and as permitted by security). The resulting planning data is saved to the database from the plan files.
- *Utilities* perform associated functions for a particular file group, such as allocations. Utilities are defined as needed for each file group—your file group may have no utilities, or several.
- *Process management* can be used to manage plan files through a set of defined planning steps. When using a *plan file process*, each plan file has an assigned owner who is responsible for the "task" of editing or reviewing the plan file for that step.

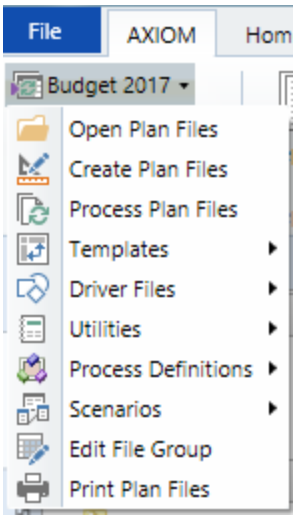



All file group components are stored in the Axiom database. Both the source files and the resulting data are stored, so that you can always see the details of how a particular plan value was calculated.

In the Axiom file system, each file group has its own folder in the **\Axiom\File Groups** folder. Within each file group folder, there is a sub-folder for each major component. Administrators can access these folders using Axiom Explorer.

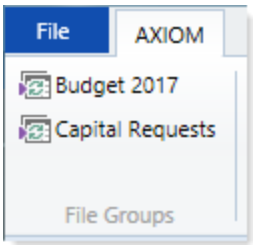
## Accessing file groups

You can access file groups from the **Axiom** tab or from the **Explorer** task pane. Using these file group buttons, you can open, create, and process plan files, and manage file group components such as templates and drivers.




By default, only administrators can see the file group administration menu for each file group. Non-admin users only see this menu if they have been specially granted permissions to certain files or features—such as permission to run Process Plan Files, or permission to access a particular driver file. In that case, the menu is limited to only showing what the user has permission to access.

If an end user does not have any special file group permissions, then that user can only see the file group if the user has permission to any plan files in that file group. The file group menu does not display in this case; instead the user sees a single item with the file group name, which can be used to open plan files in the file group.




Although the Explorer task pane displays all file groups that you have security permission to access, the Axiom ribbon tab is configurable and only displays the file groups that have been explicitly placed there for display. For example, a user might have access to four file groups (including file groups for past planning cycles) but the Axiom ribbon tab may be configured to only display the two file groups that are currently in active planning cycles. For more information on how to show file groups on the Axiom ribbon tab (or on other custom ribbon tabs), see Placing file groups on the ribbon.

## Plan code tables for file groups

When a file group is first created, it is assigned a specific reference table to be used as the *plan code table* for the file group. The plan code table holds the list of *plan codes* for the file group. For example, budgets might be created at the department level, forecasts at the division level, and capital plans at the project level.

The plan code table can be any reference table in the database, representing any list of items that you want to plan for. The same reference table can be used as the plan code table for multiple file groups. Essentially, the plan code table defines the "dimension" by which you want to plan.

Generally speaking, there is a plan file for each code in the plan code table, although there does not have to be. For example, multiple codes in the plan code table could be mapped to another code, and then all planning for those codes could be done in one plan file.

The plan code table can be a predefined list of items, such as departments, or an undefined list that is created on demand, such as ID codes for capital project requests. File groups that use undefined lists are known as on-demand file groups, and have some additional setup considerations and differing process flows than "standard" file groups. For more information, see On-Demand File Groups.

In addition to the list of plan codes, the plan code table can contain a number of other columns. These columns can be used for grouping and filtering purposes (for example: VP, Region, Manager) or to enable certain system processes. For example, you can create a column that controls whether a particular plan code will be included in file group processes. Any column can be used for these system processes; when you define the file group, you specify which column is to be used (if any).

The plan code table must already exist in order to create a file group that uses it. Once a file group has been created, you can only change its plan code table if the file group has no plan files; otherwise the setting is locked and cannot be changed.

## Using multiple file groups

Each Axiom system can have as many file groups as necessary to meet your planning needs. Multiple file groups can be used for many different purposes:

- **Differing levels or categories of planning.** For example, you might have one file group for department-level budgeting, and another file group for division-level forecasting.
- **Archival.** Once a planning cycle is complete, you can create a new file group to start a new cycle of planning, and keep the old file group as an archive. You can also clone a file group at any point in time to create an archive copy of it.
- **Scenarios and versions.** You can clone a file group to create a different version or scenario, make changes to the drivers, and compare the differences.
- **Security.** For example, you may need to restrict access to certain areas of the budget, such as payroll. You could use several file groups, all tied to the same plan code table, but where the plan files in each file group cover a different aspect of planning—payroll, capital, revenue, expenses. You could then limit user access on a per file group basis. Because plan files can query the database like a report, you could have a summary section in each plan file, summarizing the totals of the other budget areas for reference, but not revealing the detail.

# Managing file groups

Using Axiom Explorer, you can create new file groups, edit the settings of existing file groups, and delete file groups.

By default, only administrators can manage file groups. Non-admin users can be granted one of the following permissions in order to manage file groups:

- **Administer File Groups** is a global permission that allows the user to create, edit, and delete any file group.
- **Modify File Group** is a file group-specific permission that allows the user to modify the settings of a particular file group.

## Creating a file group

When creating a new file group, you can start with a "blank" file group, or you can clone an existing file group. This topic explains how to create a new blank file group. If you want to clone an existing file group, see Cloning an existing file group.

This approach creates a new file group with no settings and no files. You will need to complete all file group settings, and then create or import the necessary files such as templates and driver files.

Before creating a new file group, make sure that the associated plan code table already exists in Axiom. You must specify a plan code table in order to save the file group.

**NOTE:** Only administrators or users with the **Administer File Groups** security permission can create new file groups.

**To create a new file group:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Right-click the **File Groups** header (or any existing file group), and then select **New > File Group**.

3. In the **Create New File Group** dialog, complete the following basic settings for the file group.

| Item | Description |
|---|---|
| File Group Name | The full name of the file group. |
| File Group Year | Optional. The year of planning for the file group.<br><br>Click **use in file group name** if you want this year added to the file group name as a variable (for example, "Budget 2015"). This option only displays if you have defined a year value. The full resolved name then displays in gray text to the right of the box. |

| Item | Description |
|---|---|
| File Group Category | Optional. Assign the file group to a category for display on the ribbon and in the Explorer task pane. |
| Plan Code Table | The reference table that defines the list of plan codes for the file group. |
| Display Name | The display name for the file group, to display in the Axiom ribbon tab, Axiom Explorer, and other areas where file groups are listed. By default the file group name is used as the display name unless you define a different display name. |
| Tab Prefix | Optional. A brief code to display on file tabs for files in the file group, to identify the file as belonging to that file group. |

**NOTE:** You can use any file group variable in the File Group Name, Display Name, and Tab Prefix. However, if you want to do this then you must first create the file group with temporary names, because it is not possible to create file group variables in this initial creation screen. Once the file group has been created, then you can access the full file group properties, create the variable, and then change these settings to use the variable as desired.

4. Click **OK** to create the new file group.
5. A confirmation message informs you that the new file group was created successfully. Click **OK** to continue.
6. The **Edit File Group** dialog opens. You can now edit any setting for the new file group. When you are finished, click **OK**.

**NOTE:** On the **Options** tab, you must specify either a **Default Template** or a **Template Column** before you can create plan files for this file group. You can edit this setting later if you are not ready to configure it now.

For more information on all file group properties, see File Group Properties.

**NOTES:**

- When a new blank file group is created, no non-admin users can access it. You must configure access rights to the new file group within Security.
- If the new file group is not assigned to a category, then it will not display on the Axiom ribbon tab (or any custom ribbon tab) unless you edit the ribbon tab to place it there.
- If the new file group is assigned to a category, and the category is configured to display on the Axiom ribbon tab (or any custom ribbon tab), then by default the new file group will display on the ribbon within the category, as the last file group in the category list. If you want to change the order of the new file group, you must use **Organize File Group Category** to change the configuration.

## Cloning an existing file group

You can clone an existing file group in order to create a new file group. Typically, file groups are cloned for one of the following reasons:

- To create a new file group to use for a new cycle of planning
- To create an archive copy of the file group

As part of the cloning process, you can specify which file group components (templates, drivers, etc.) are copied to the new file group. Keep in mind that any copied files may need to be edited for use in the new file group (unless you are cloning the file group for archive purposes). The cloning process can also automatically create tables for the new file group, if the necessary tables do not already exist.

**NOTE:** Only administrators or users with the **Administer File Groups** security permission can clone file groups. If new tables are to be created as part of the cloning process, then the user performing the process must also have the **Administer Tables** security permission.

**To clone an existing file group:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. In the **Axiom Explorer** dialog, select the file group that you want to clone, then click **Clone** in the Axiom Explorer toolbar.

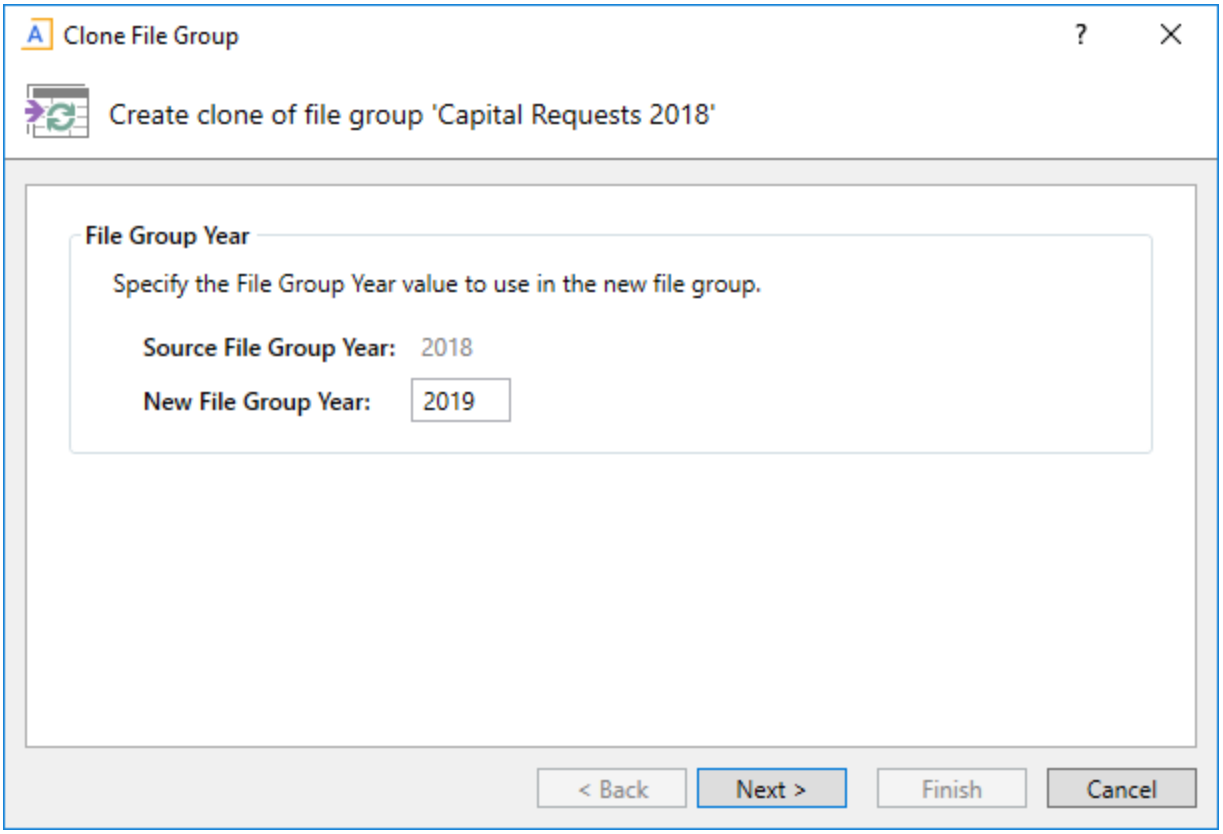
   **TIP:** You can also access this command from the right-click menu, in the Axiom Explorer dialog and in the Explorer task pane.

The **Clone File Group** dialog opens to guide you through the cloning options. Complete each screen as needed, and then click **Finish** to create the new file group.

The choices that you make in each screen depend on the file group configuration and on the reason you are cloning the file group. See the following sections for more information on each screen, and for details on what happens when the new file group is created.

## File Group Year

If the file group has a defined file group year, specify the **New File Group Year** for the new file group, or leave the year the same.




*Example File Group Year screen*

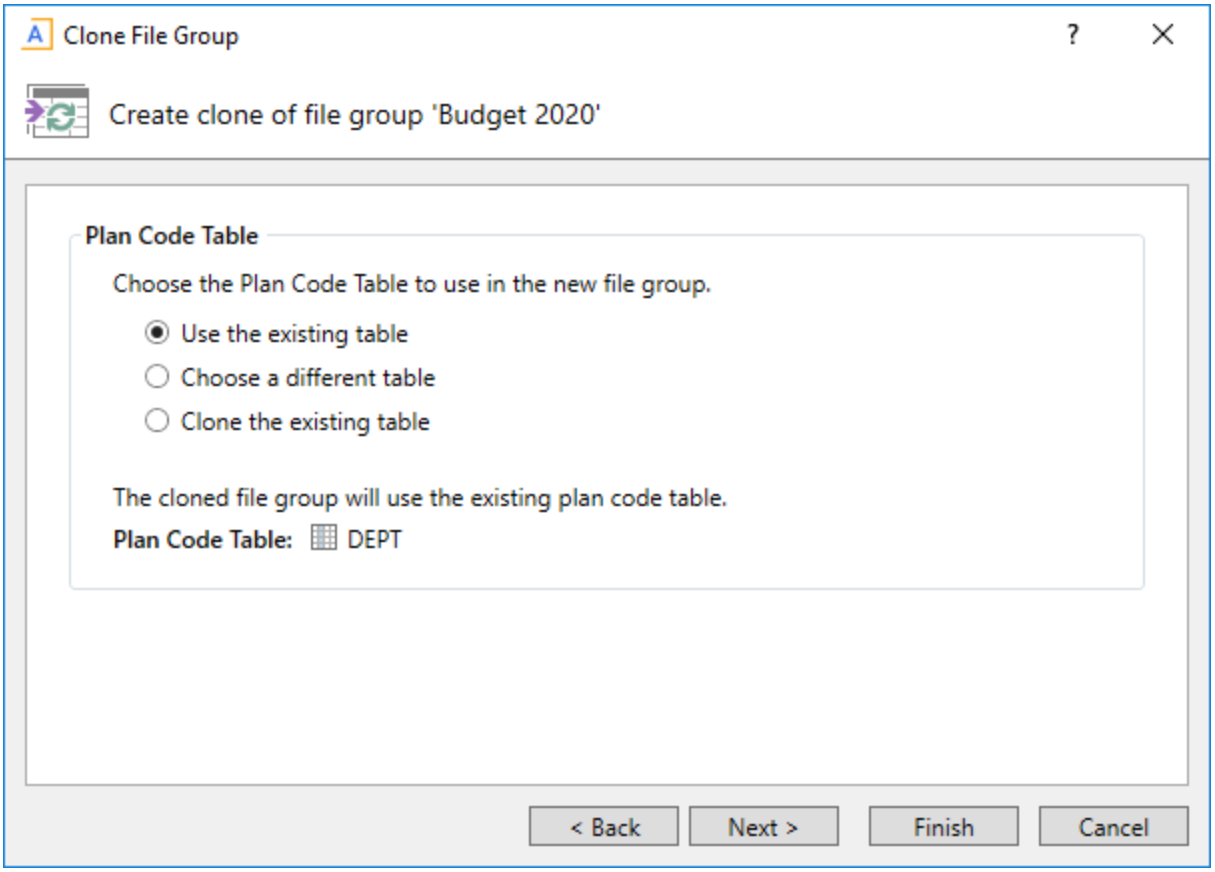
By default, the new file group year is set to the same year as the original file group year.

- If you are creating a new file group to use for a new cycle of planning, then you should change the file group year to reflect the next planning period.
- If you are creating an archive copy of the file group, then you should leave the file group year as is.

If the original file group does not have a defined file group year, then this screen does not display and instead the wizard begins with the Plan Code Table screen.

## Plan Code Table

Specify the plan code table to use for the new file group.




*Example Plan Code Table screen*

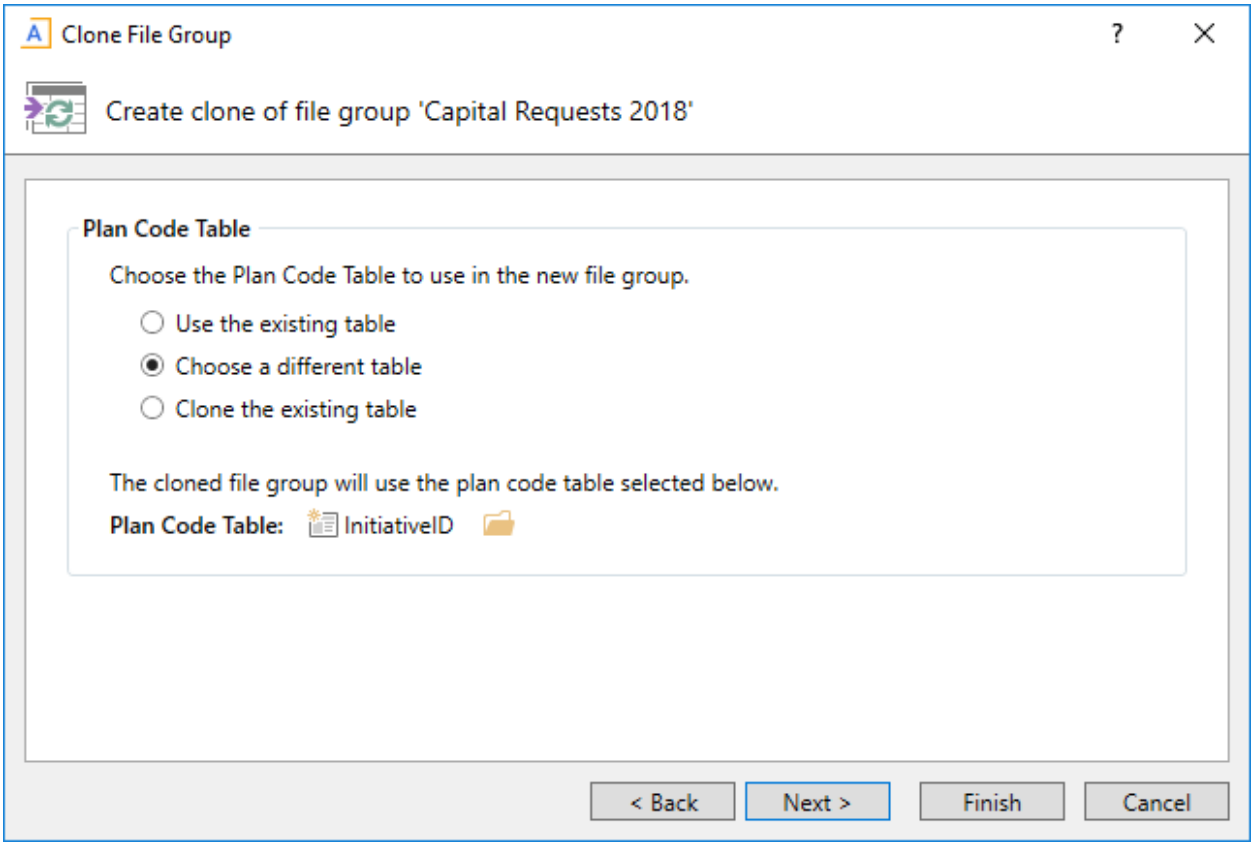
You can choose from the following options:

| Option | Description |
|---|---|
| Use the existing table | The new file group will use the same plan code table as the original file group. Use this option if you are creating an archive copy, or if the file group uses a fixed list of predefined codes. |
| Choose a different table | Select a different, existing table to use for the new file group. This option is not commonly used, but it can come in handy if the file group needs to use a new file group table that has already been created. |
| Clone the existing table | Clone the existing plan code table and use the new table for the new file group. Only the table structure will be cloned, not the data.<br><br>This option is most commonly used when cloning an on-demand file group to start a new planning cycle, to create a new identity table with the same structure as the old identity table. For example, if the original file group uses CapReq2021, you want to clone that table and name it CapReq2022. |

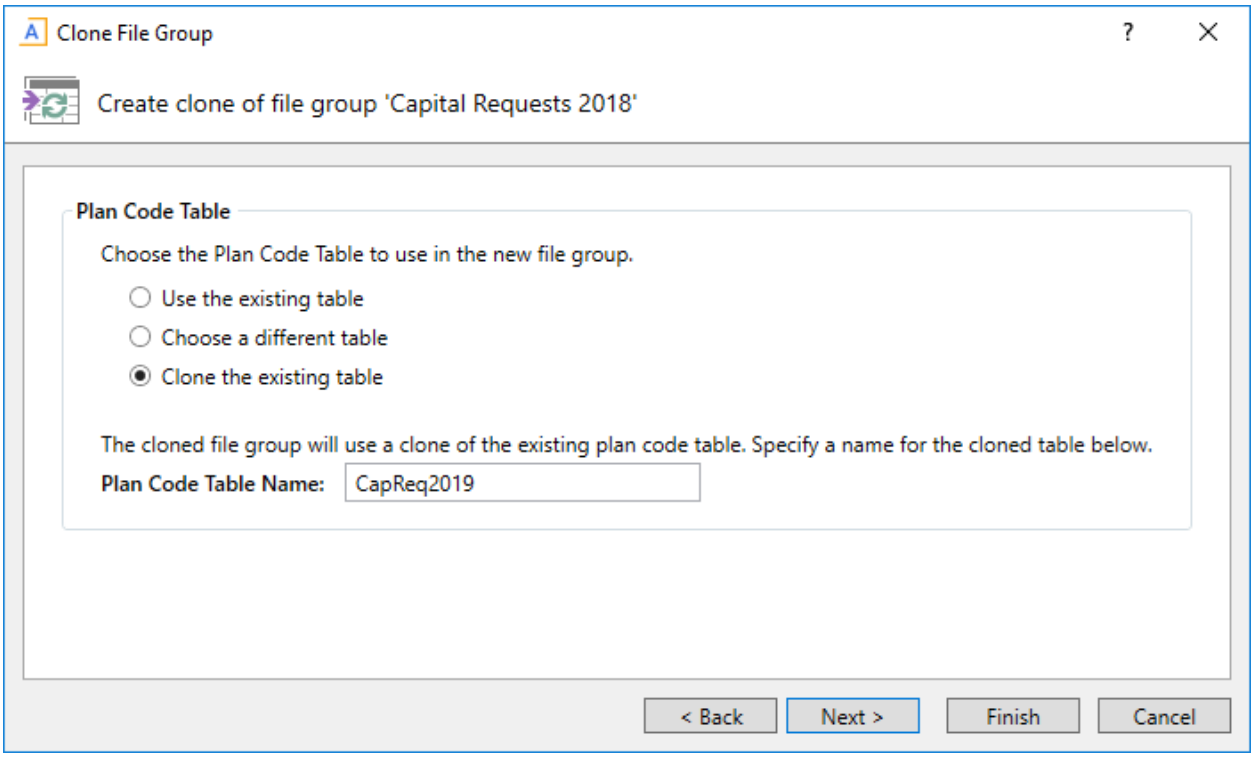
Axiom attempts to intelligently populate this screen depending on the name of the plan code table and the state of the file group year:

- If the name of the existing plan code table contains a year that matches the file group year, and you change the file group year as part of the cloning process, then Axiom assumes that you want to use a new plan code table that contains the new year.
    - If that table already exists, then **Choose a different table** is automatically selected and points to the existing table.
    - If that table does not automatically exist, then **Clone the existing table** is automatically selected and the new table name is predefined using the new year.
- If the name of existing plan code table does not contain a year, or if it does but you did not change the file group year, then **Use the existing table** is automatically selected.
- You can modify this screen to choose different options if the automatic selection does not meet your needs.

If **Choose a different table** is selected, you can click the folder icon at the bottom of the screen to browse to the desired table.




If **Clone the existing table** is selected, you can enter a name for the new table into the **Plan Code Table Name** box.

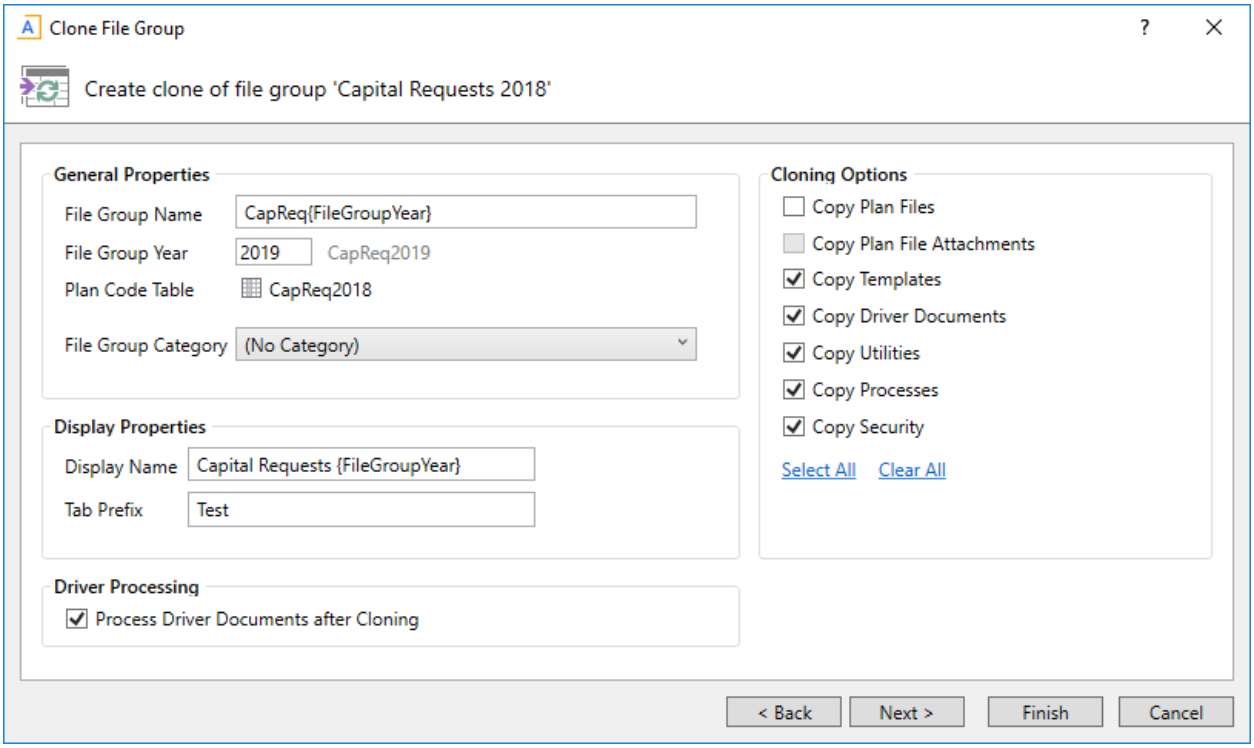



If the plan code table is changed as part of the file group cloning process (either by choosing a different table or by cloning the existing table), Axiom attempts to update settings that point to the plan code table as follows:

- When the new file group is created, Axiom attempts to update various column settings inherited from the original file group, so that these settings now use corresponding columns in the new plan code table. For example, if the original file group uses a column named AltCode as the **Tab Column**, and the new plan code table also has a column named AltCode, then the Tab Column setting for the new file group is automatically updated to use the column in the new table. However, if the new plan code table does not contain a matching column, then the setting is cleared in the new file group and you must reconfigure it in the file group properties.
- If other tables are also cloned (via table variables), and those tables have columns that look up to the plan code table, Axiom attempts to update these lookups to the new plan code table. For example, if table CapData2021 has a lookup to `CapReq2021.CapReq`, and the new plan code table is CapReq2022, the lookup will be updated to `CapReq2022.CapReq` if possible.
- If **Copy Security** is enabled for the file group cloning process (in the **Cloning Options** screen), then when the plan file permissions are copied to the new file group, Axiom attempts to update the security filters to point to the new plan code table. If the new plan code table does not have the column used for a particular security filter, then the copied filters are invalid and must be manually changed.

## File Group Properties and Cloning Options

Specify various properties for the new file group, and specify cloning options to determine what is copied to the new file group.




*File Group Properties and Cloning Options*

### General Properties

The primary property to review in this section is the **File Group Name**, which must be different than the original file group's name.

| Item | Description |
| --- | --- |
| File Group Name | Specify the name of the new file group. The original file group's name is shown by default.<br><br>If the name uses a variable, then you must change the value of the variable so that the resolved name is different than the original file group's name. If the name uses the `{FileGroupYear}` variable, and you changed the year earlier in the wizard, then the name will resolve using the new year and no further action is necessary. However, if you are creating an archive copy and did not change the year, then you must change the name to something like `Budget {FileGroupYear} Archive`.<br><br>If the name does not use a variable, then you must manually change the name to something different. |
| File Group Year | This setting is repeated on this screen just to provide context. If you changed the file group year in the first screen of the wizard, the new value is shown here. |
| Plan Code Table | This setting is repeated on this screen just to provide context. It displays the table selection from the previous screen. |
| File Group Category | If the original file group was assigned to a category, this category is retained by default for the new file group. You can modify the setting as desired or leave the default value.<br><br>**NOTE:** If the new file group belongs to a category, and that category is configured to display on the Axiom ribbon tab (or any custom ribbon tab), then the new file group will display on the ribbon as soon as it is created (for users with access to the new file group). If you do not want the file group to display on the ribbon after creation, you must remove it from the category. |

### Display Properties

Review display properties for the new file group.

| Item | Description |
| --- | --- |
| Display Name | By default, this is the same display name as defined for the original file group. If the display name is the same as the file group name, then it will automatically update for the changed file group name. You can modify the setting as desired or leave the default value. |
| Tab Prefix | By default, this is the same prefix as defined for the original file group. You can modify the setting as desired or leave the default value. |

### Driver Processing

Determines whether driver documents are processed as part of the file group clone.

| Item | Description |
|---|---|
| Process Driver Documents After Cloning | Specifies whether copied driver files in the new file group will be processed automatically after the file group is created. If enabled, the driver files will be opened, calculated, and saved. The save includes both a save-to-database and a file save. Axiom queries and data lookups are only run if they are set to refresh on open.<br><br>This option is primarily intended for file groups that use Save Type 3 drivers. This process will create the document reference tables for the new file group. You should be sure that the driver table names are unique and will not overwrite the tables from the original file group. Ideally these names should be determined by use of table variables in the file group.<br><br>For more information, see Drivers and file group cloning. |

### Cloning Options

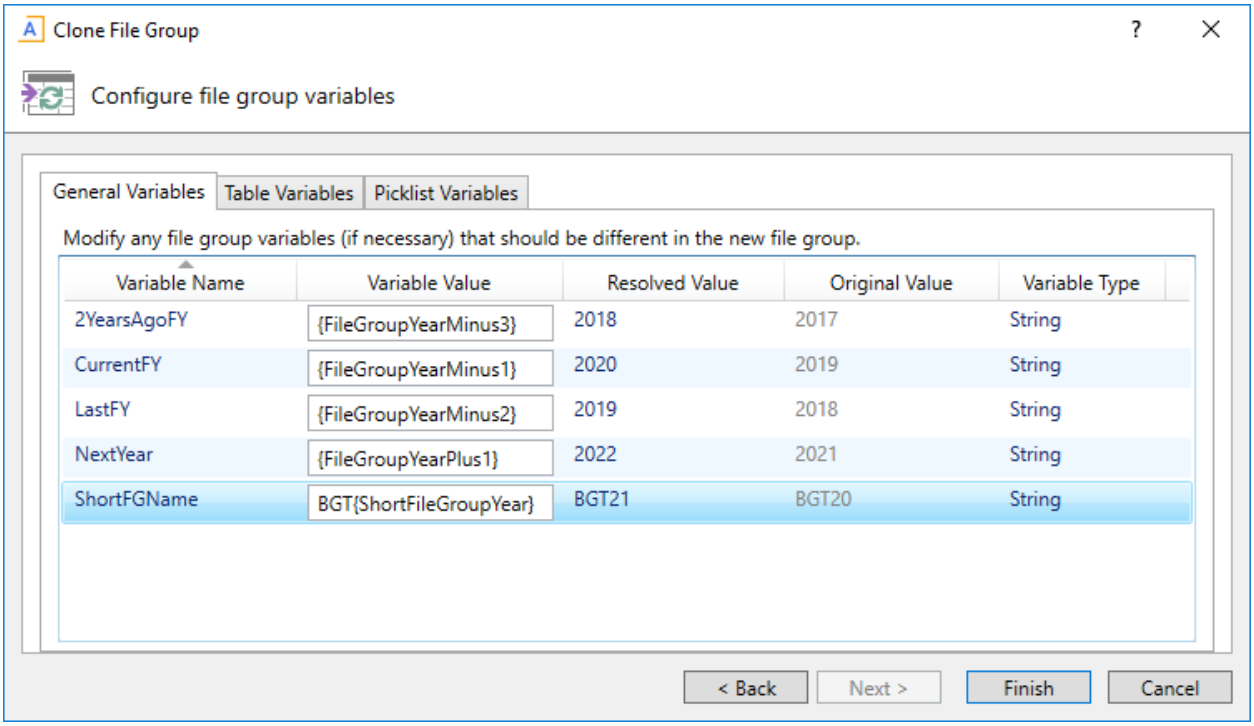
Specify cloning options for the file group, to determine which file group entities are copied to the new file group.

| Item | Description |
|---|---|
| Copy Plan Files | Select this option to copy plan files from the original file group to the new file group.<br><br>Typically the only reason to copy plan files is if you are creating an archive copy. If you are cloning a file group to start a new cycle of planning, then you will create new plan files from a template and do not need to copy plan files.<br><br>**NOTE:** If the file group uses virtual plan files, then enabling this option will cause the placeholder document records to be copied to the new file group, so that they can be used in the same way within the new file group. The physical plan files do not exist and therefore cannot be copied. |
| Copy Plan File Attachments | Select this option to copy plan file attachments from the original file group to the new file group.<br><br>Typically the only reason to copy plan file attachments is if you are creating an archive copy. If you are cloning a file group to start a new cycle of planning, then you will create new plan files from template and do not need to copy attachments.<br><br>This option is only available if Copy Plan Files is selected, and if plan file attachments are enabled for the file group. |

| Item | Description |
| --- | --- |
| Copy Templates | Select this option to copy templates from the existing file group to the new file group. The associated calc method libraries are copied along with the templates. |
| Copy Driver Documents | Select this option to copy driver files from the existing file group to the new file group. |
| Copy Utilities | Select this option to copy utilities from the existing file group to the new file group. |
| Copy Processes | Select this option to copy process definitions from the existing file group to the new file group.<br>• If a process is active when it is copied, it will be made inactive in the new file group.<br>• If the Plan File Process setting was specified for the cloned file group, it will be updated to point to the appropriate copy of the process in the new file group. |
| Copy Security | Select this option to copy security settings from the existing file group to the new file group. Keep in mind the following:<br>• If you do not copy security settings, then non-admin users will not have rights to the file group when it is created. You will need to manually configure access to this file group.<br>• If you do copy security settings, then any user or role who has rights to the existing file group will have equivalent rights to the new file group, as soon as the file group is created.<br>File security settings for templates, drivers, utilities, and processes are only copied if you chose to copy those files. Otherwise, only the settings on the File Groups tab are copied.<br>**IMPORTANT:** If the plan code table was changed as part of the cloning process and Copy Security is selected, then Axiom updates the existing filters to point to the new plan code table. If the new plan code table does not have the column used for a particular security filter, then the copied filters will be invalid and must be manually changed.<br>**NOTE:** If Copy Security is selected and tables are created as part of the cloning process, then security settings will be copied from the original tables to the new tables. |

## Variables

Review the file group variables and make any necessary changes for use in the new file group. For each variable, you can see the new resolved value and the original value for comparison. Make sure to review all three tabs: **General Variables**, **Table Variables**, and **Picklist Variables**.




*Example File Group Variables screen*

If you have variables that reference the file group year, these variables are already updated as needed based on the file group year that you changed earlier in the wizard. Any other necessary changes must be made manually.

You can only modify the value of the existing variables when cloning a file group; you cannot add or remove variables. Once the new file group has been created, you can make any further changes to the variables as needed.

**NOTES:**

- Cloned tables are placed in the same folder as the original tables. If you want any of the new tables for the new file group to reside in different folders, you must manually move the tables to the appropriate folders after the new file group is created.
- Document reference tables are not cloned, because they are sourced from a file. Instead, the driver files can be cloned and then processed to create the new document reference tables in the new file group. It is still recommended to use table variables to set the names of the document reference tables, so that they can automatically update for the new file group.

### Special considerations for table variables and picklist variables

When cloning the file group to start a new year of planning, the table variables should resolve to different tables in most cases. For example, if the original file group queried data from GL2020 and saved data to BGT2021, then the new file group may need to query data from GL2021 and save data to BGT2022. If you are cloning the file group to create an archive, then the table variables should resolve to the same tables.

If a table variable or a picklist variable resolves to a table name that does not exist, the cloning process will create that table as part of creating the new file group. For example, if the table variable originally resolved to BGT2021 and now it resolves to BGT2022, table BGT2022 will be created by cloning BGT2021. This table cloning occurs as follows:

- For data tables, only the table structure is copied. Data is not copied.
- For reference tables (including picklist tables), both the table structure and data are copied. Exception: If the key column of the reference table is an identity column, the data is not copied.
- In both cases, if **Copy Security** is enabled on the Cloning Options screen, security is copied to the new table.

## Creating the new file group

When you click **Finish** to create the new file group, the following occurs:

- If any of your table variables that allow saving data resolve to existing tables, a warning message displays and lists these tables. You should review this list and make sure that these are the intended tables for this file group. If they are not the intended tables, then click **Cancel** to return to the Clone File Group dialog and adjust your table variables. If they are the intended tables, click **OK** to continue.
- If any of your table variables or picklist variables resolve to data tables or reference tables that do not yet exist, then a confirmation message will inform you that these tables will be created. Click **OK** to continue.

If you opted to process drivers after the new file group has been created, this process will occur automatically after all files have been copied and any new data tables have been created.

The **Edit File Group** dialog opens. If desired, you can now edit any setting for the new file group. By default, the new file group inherited all settings that were not addressed in the Clone File Group dialog from the original file group. For more information, see File Group Properties.

**NOTES:**

- If Triggers were defined in the original file group, they are inherited by the new file group but they are disabled by default. You should review the triggers to determine whether they should be edited for the new file group and re-enabled, or deleted as unnecessary.
- If an Add File Form was defined for this file group, the setting is retained if the assigned file is a file group utility and utilities were copied as part of the clone. If the form was configured to **Use Current File Group** then it will automatically be updated for the new file group; if not then you must edit the form to point to the new file group. However, if the assigned file was a report file in the Reports Library, then the Add File Form setting is now cleared in the new file group and you must re-assign it as appropriate.

### Drivers and file group cloning

The file group cloning process handles driver tables differently depending on what kind of tables they are. Driver tables can be managed in two different ways:

- Driver files can use Save Type 3 to create *document reference tables*. In this case the data and table structure is managed within the file. The table is tied to the file and can only be modified by the file.
- Driver files can use Save Type 1 to modify the data held in *reference tables*. In this case the table exists independently from the driver file, and can be managed as a normal table. The driver file simply provides a means to modify the data.

In both cases, the driver tables should be defined as writeable table variables (meaning **Allow file group to save data to this table** is enabled). If you want new driver tables to be created as part of the file group cloning process, then these table variables should resolve to new names for the new file group. However, the way in which these tables are created differs.

When driver tables are document reference tables, the tables are not copied directly by the file group cloning process. Instead, the driver files are copied and processed, which creates the new document reference tables for the new file group. Because data is managed in the file, the data is copied by copying the driver files, not by copying data between tables. You can process the driver files automatically as part of the cloning process, or by saving them manually after the new file group is created.

When driver tables are reference tables, the tables are copied by the file group cloning process. To bring the driver data to the new file group, the data in the reference tables is copied as well as the table structure. You can still choose to process the driver files if desired as part of the cloning process, but in this case it is not necessary to do this to create the new tables.

## Editing file group properties

All file group properties can be edited, with the following exceptions:

- The file group ID and code are system-generated and cannot be changed.

- The plan code table can only be changed if no plan files have been created for the file group. If the plan code table cannot be changed, this may impact the availability of other settings that depend on the plan code table.
- If you change the file group name, this changes the file paths for the file group. Alternatively, you can leave the file group name as is and instead change the display name only.

Only administrators and users with one of the following security permissions can edit file group properties: **Administer File Groups** and **Modify File Group**.

**To edit file group properties:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.
2. Locate the file group that you want to edit, then right-click it and select **Edit**.
3. In the **Edit File Group** dialog, make any edits as desired, and then click **OK**.

## Deleting a file group

Deleting a file group deletes all files stored in the associated file group folder, including templates, plan files, drivers, calc method libraries, and process definitions.

Additionally, any document reference tables that are linked to the file group are also deleted. Generally this means document reference tables that were created from driver files by using Save Type 3. If the driver files save to regular reference tables using Save Type 1, those tables will not be deleted.

A file group cannot be deleted if it is assigned to a file group alias. In this case, you must first edit the alias to point to another file group (or delete the alias) before the file group can be deleted.

**NOTE:** Only administrators or users with the **Administer File Groups** security permission can delete file groups.

**IMPORTANT:** Deleting a file group cannot be undone, and the deleted files cannot be recovered using normal Axiom functionality. You should be sure that you no longer need the file group before you delete it. You may want to take a backup of the Axiom database before deleting the file group.

**To delete a file group:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.
2. Navigate to the file group that you want to delete, then right-click the file group and select **Delete**.
3. At the confirmation prompt, click **Yes**.

The file group and all associated files are deleted.

# Using file group aliases

You can define alias names for file groups, so that a file group can be referenced by the alias name instead of the file group name. This feature is intended to support the rollover process and other configurations that require file groups to be dynamically referenced.

For example, you can create an alias named "Current Budget," and assign that alias to the Budget 2021 file group. When setting up file group references in components such as custom ribbon tabs or Scheduler jobs, you can point these items to the Current Budget alias instead of directly to the Budget 2021 file group. Then when you are ready to start a new year of planning, you can edit the Current Budget alias so that it is now assigned to the Budget 2022 file group. All components that reference the Current Budget alias will now point to the Budget 2022 file group instead of the Budget 2021 file group.

## When to use file group aliases

If you perform rollover by cloning the current file group and then using the new file group for the new cycle of planning, it is recommended to use file group aliases for components that are external to the file group, to make the rollover easier. First you would clone the current file group, and then test and update the new file group as needed. When you are ready to roll out the new file group to end users to start the new cycle of planning, you would then update the appropriate alias to point to the new file group.

If you do not use aliases, then you would have to identify all affected components and then manually update each component to point to the new file group. For example, if you have a Scheduler job that runs Process Plan Files for the budgeting file group on a regular basis, as part of your rollover you would need to update this job so that it points to the new file group. If instead the job uses an alias, then the job will update automatically when the alias is changed.

Although aliases can be quite useful, there are certain circumstances where you should not use aliases:

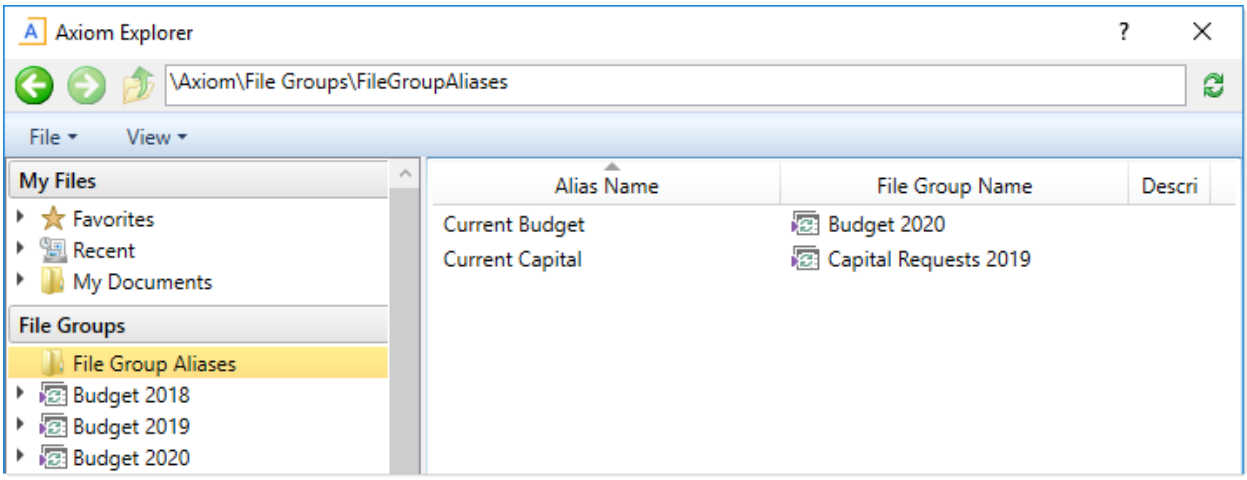
- If your planning cycles overlap, where both file groups need to be active at the same time, then aliases should not be used. "Active" means that end users need to access both file groups, processes still need to be performed on both file groups, etc. For example, if Budget 2021 and Budget 2022 need to be active at the same time, then a Current Budget alias is likely not useful because instead you would need separate Scheduler jobs for each file group, separate ribbon tab items for each file group, etc. However, if Budget 2021 is mostly inactive by the time you roll out the Budget 2022 file group, and it only needs occasional access by administrators and other power users, then aliases may still be useful.
- Components that are stored *within* a file group should not use aliases, because these components need to remain tied to the file group where they are located. For example, if you are setting up a file group utility as an Add File Form for a file group, then the Add Plan File command in that utility should use the **Use Current File Group** setting instead of pointing to a file group alias. This will keep the command pointed to the file group where the utility is located, even if you copy the file to a new file group or clone the file group. If you were to use a file group alias in this case, then you could have a situation where the Add File Form in the Budget 2021 file group now points to the Budget 2022 file group.

## Creating, editing, and deleting aliases

File group aliases are managed in Axiom Explorer, using the File Group Aliases folder. To access this area:

- On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

Select the **File Group Aliases** folder in the left-hand pane to view a list of the current aliases and file group assignments in the right-hand pane.




Only administrators and users with the **Administer File Groups** permission can create, edit, or delete file group aliases.

**TIP:** You can also create, edit, and delete file group aliases from the Explorer task pane. However, the task pane does not provide a view where you can see all of the current variable assignments.

**To create a file group alias:**

1. Right-click the **File Group Aliases** folder, and then click **New File Group Alias**.
2. In the **Edit File Group Alias** dialog, complete the following and click **OK**.

| Item | Description |
|---|---|
| Alias Name | The name of the alias. This is the name you will use when you want to reference the file group that is currently assigned to the alias.<br><br>**NOTE:** Alias names should not duplicate any file group names. If an Axiom function (such as GetFileGroupID) uses a name that can be resolved as either an alias name or a file group name, it will be resolved as the file group name. |

| Item | Description |
|---|---|
| File Group | The file group that is currently assigned to the alias. When a component—such as a task pane—references this alias, the component will actually point to the currently assigned file group. If the file group assignment changes, the component will point to the new assignment. |
| Description | Optional. A description of the alias. This is for administrative use only, to explain the purpose of the alias. |

The alias is created and is now available for use.

**To edit a file group alias:**

1. In the **File Group Aliases** folder, double-click the alias that you want to edit.
2. In the **Edit File Group Alias** dialog, edit the alias properties as desired and then click **OK**.
   - The most common edit is to change the file group assignment for the alias, such as part of a rollover process.
   - Use caution before editing the alias name. Although shortcuts to the alias will automatically update, if the alias name is used in Axiom functions such as GetFileGroupID, those functions must be manually edited for the new name.

The alias is saved with the edits.

**To delete a file group alias:**

1. In the **File Group Aliases** folder, right-click the alias and then click **Delete**.
2. At the confirmation prompt, click **Yes** to continue.

Any components that used to reference this alias will no longer work.

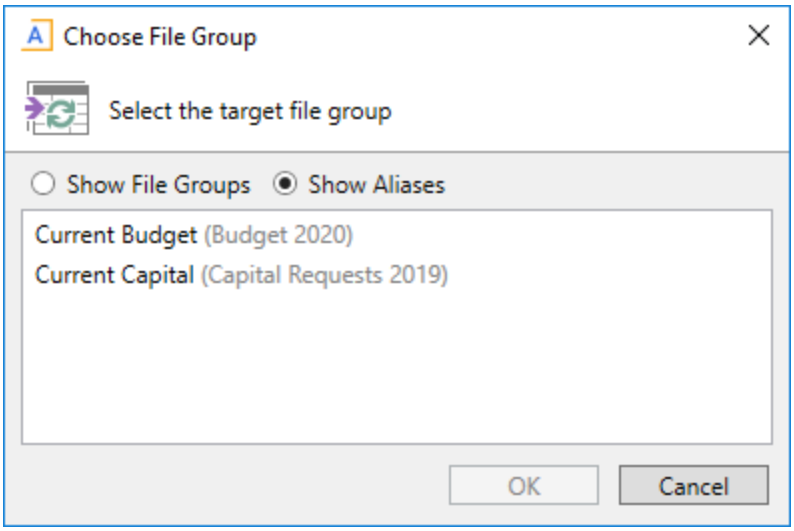
## Referencing file groups using the alias name

You can use an alias name to reference a file group in various features. For example:

- Custom task panes and ribbon tabs — Use aliases as shortcut targets, and in shortcut parameters for commands such as Add Plan File. You can also point to specific files and folders in a file group using aliases.
- Commands — Use aliases in shortcut parameters for commands such as Add Plan File.
- Scheduler jobs — Use aliases as the designated file group for tasks such as Process Plan Files, and specify files to process via an alias for tasks such as Process Document List.
- Axiom functions — Use aliases as the file group name in functions such as GetFileGroupID.

### Selecting a file group using an alias

For features where you select a file group within a dialog, you can now choose to use either a file group name or an alias name. Select **Show aliases** to choose an alias name. The current file group assignment for the alias is displayed in parentheses after the file group name.




### Using a file group alias in an Axiom function

For Axiom functions that take a file group name or a file group ID, you can substitute the alias name for a file group name. For example:

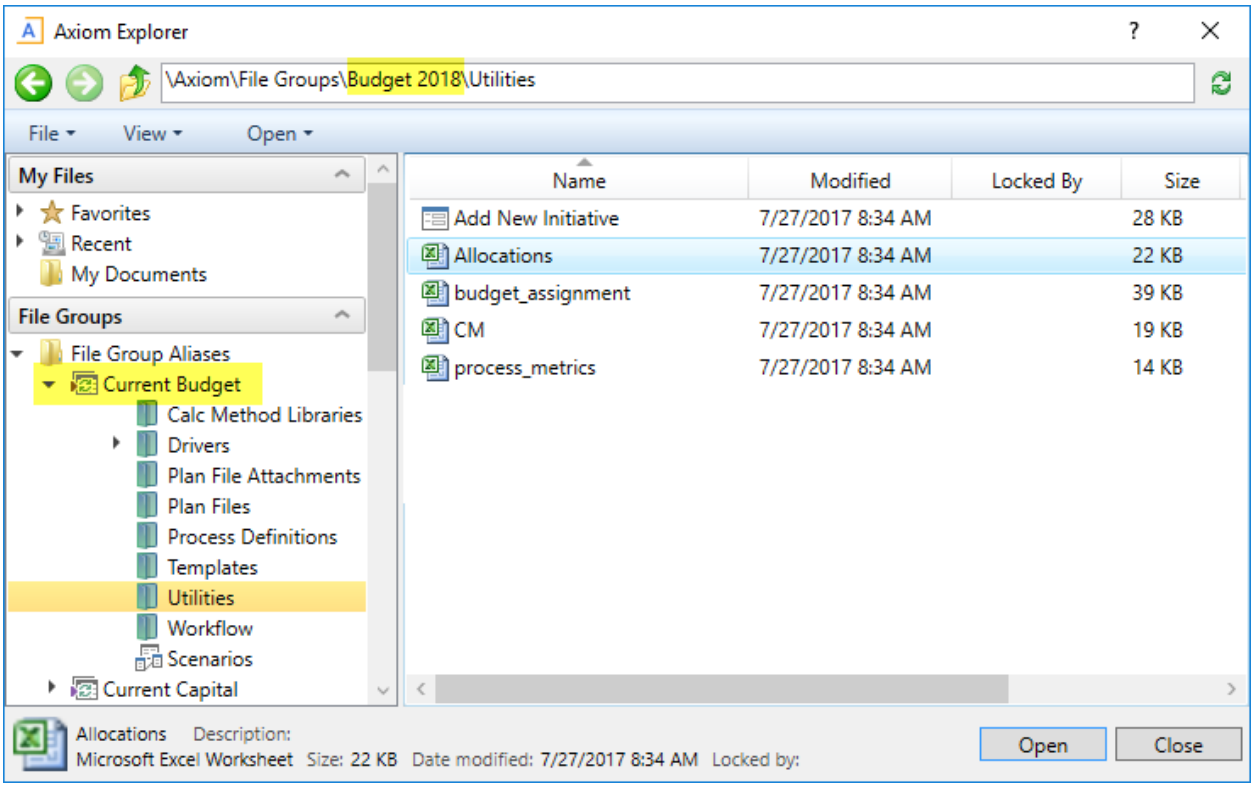
```
=GetFileGroupID("Current Budget")
```

Where Current Budget is an alias name that is currently assigned to the Budget 2022 file group. The function will return the ID for the Budget 2022 file group. The ID can then be used in other functions such as GetFileGroupProperty or GetFileGroupVariable.

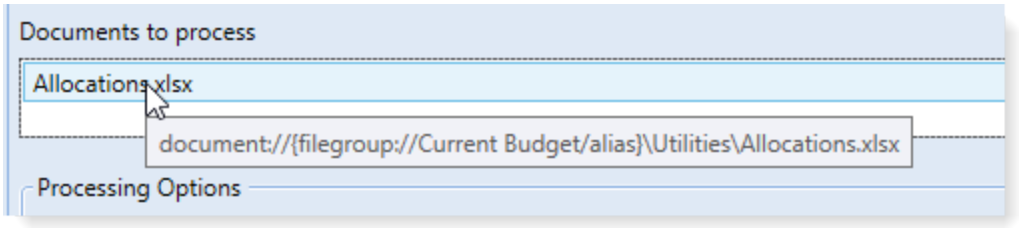
### Selecting a file using an alias

Some features allow you to select a file or folder using the file group alias name, so that the path to the file or folder is written using alias syntax. This makes the reference dynamic so that the feature will automatically look for the file or folder within the current target of the file group alias.

For example, when choosing a document for the Process Document List task, you can navigate to the document through the alias:




The path to the document is then written using alias syntax:




This document selection via file group alias is available for Scheduler tasks Process Document List and Process Template List, in task panes and ribbon tabs for document and folder shortcuts, and for the Show Form Dialog command.

# Using file group categories

You can use file group categories to group certain file groups together and display them as a unit on the ribbon and in the Explorer task pane.

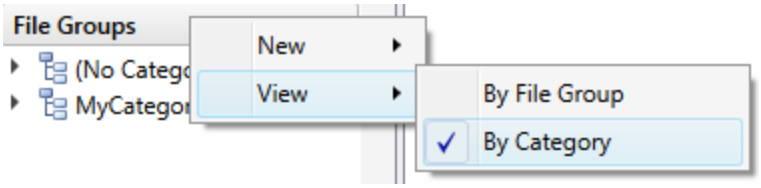
File group categories are for display purposes only. No file group or security settings are impacted by use of categories.

**To view and work with file group categories:**

1. On the **Axiom** tab, in the **Administration** group, select **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** header.

2. Right-click the **File Groups** header, and then select **View > By Category**.




**TIP:** You can also change the view from the Explorer task pane.

File groups are now displayed by categories. File groups that do not belong to a category are organized under **(No Category)**.

## Managing categories

- To add a new category, right-click the **File Groups** folder (or an existing category) and then select **New File Group Category.** You then enter a name for the category, and an optional description.
- To edit a category, right-click the category and select **Edit File Group Category**. You can edit the name and description.

  Alternatively, if you just want to change the name of the category, you can select **Rename** from the right-click menu.
- To delete a category, right-click the category and select **Delete**. You cannot delete a category if file groups are assigned to it.

## Assigning file groups to categories

To assign file groups to categories, you can drag and drop file groups to the desired categories, or you can assign the category within the file group properties (right-click the file group and select **Edit File Group**).

## Category display to users

If a user has access to any file groups in a category, then that category will display in the user's Explorer task pane. The user will only see the file groups that they have access to; any other file groups in the category will be hidden. If the user does not have access to any file groups in the category, then the category will not display. Administrators have access to all file groups and will see all categories.

If you want a file group category to display on the Axiom ribbon tab (or on any other custom ribbon tab or task pane), then you must edit the ribbon tab to place the category on the tab. For more information, see Placing file groups on the ribbon.

If desired, you can specify the display order for file groups in a category. This display order only applies when the category is placed on a custom ribbon tab or task pane.

**To define the display order for a category:**

1. Right-click the category and then select **Organize File Groups in Category**.

   **TIP:** You can also open this dialog from the Axiom Explorer toolbar.

   The **Organize Category** dialog opens. This dialog displays all file groups that are assigned to the category, in their current display order.

2. To change the order, select the file group that you want to move and then use the arrow buttons at the top of the dialog.
3. Click **Apply** or **OK** to save.

# Placing file groups on the ribbon

If you want end users to be able to access file groups on the ribbon, you must edit the Axiom ribbon tab to place them there. There is no automatic way to display file groups on the ribbon; simply granting a user permission to a file group does not place it on the ribbon (though the file group will display in the user's Explorer task pane).

File group visibility works as follows:

- If a user does not have any security permissions to a file group, then by default the user does not see that file group in ribbon tabs, task panes, or other menus.
- If a user is granted security permissions to a file group, then the file group will display automatically in the user's Explorer task pane. Administrators see the full list of file groups in the Explorer task pane. By default, the view in the task pane is set to category. All users have the option to change the view by right-clicking the File Groups header.
- If a file group is placed on a custom ribbon tab or task pane, then a user will see it if that user has security permissions to that file group. Otherwise, it will be hidden (unless **Show restricted item** is enabled, in which case it will display as grayed out). Administrators always see a file group if it is placed on a custom ribbon tab or task pane.

You may be using the default Axiom ribbon tab (`AxiomMain.AXL`), or you may have created one or more copies of that tab that you have assigned to certain users and roles. You may also have created completely custom ribbon tabs or task panes that you expect users to use to access their file groups. This topic will discuss editing `AxiomMain.AXL` as an example of how to edit a custom ribbon tab (or a task pane) to display file groups.

For more information on working with custom ribbon tabs, see the *System Administration Guide*.

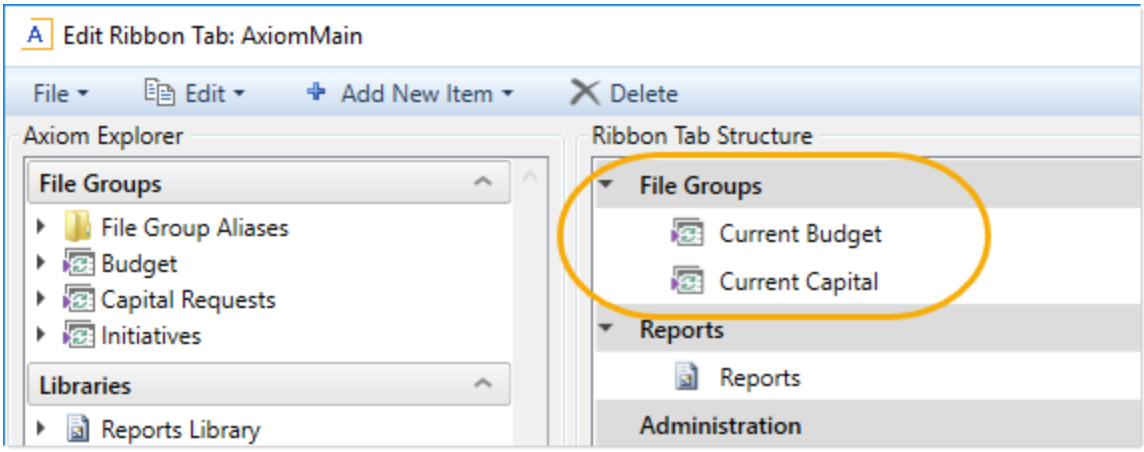
**To place file groups on the default Axiom ribbon:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > Ribbon Tabs**.

   The **Axiom Explorer** dialog opens, with the focus on the **Ribbon Tabs Library** folder. You can also use the Explorer task pane to access this folder.

2. Locate the file named `AxiomMain.AXL` and double-click it to open it for editing.

3. In the **Edit Ribbon Tab** dialog, in the **Ribbon Structure** section, locate the File Groups group. It should look something like the following example:




4. In the **Axiom Explorer** section in the left-hand side of the dialog, locate the file group or file group alias that you want to add to the ribbon, then drag and drop it as a child item in the File Groups section. For file groups, make sure to drag and drop the parent file group name, not any of the child items underneath the name.

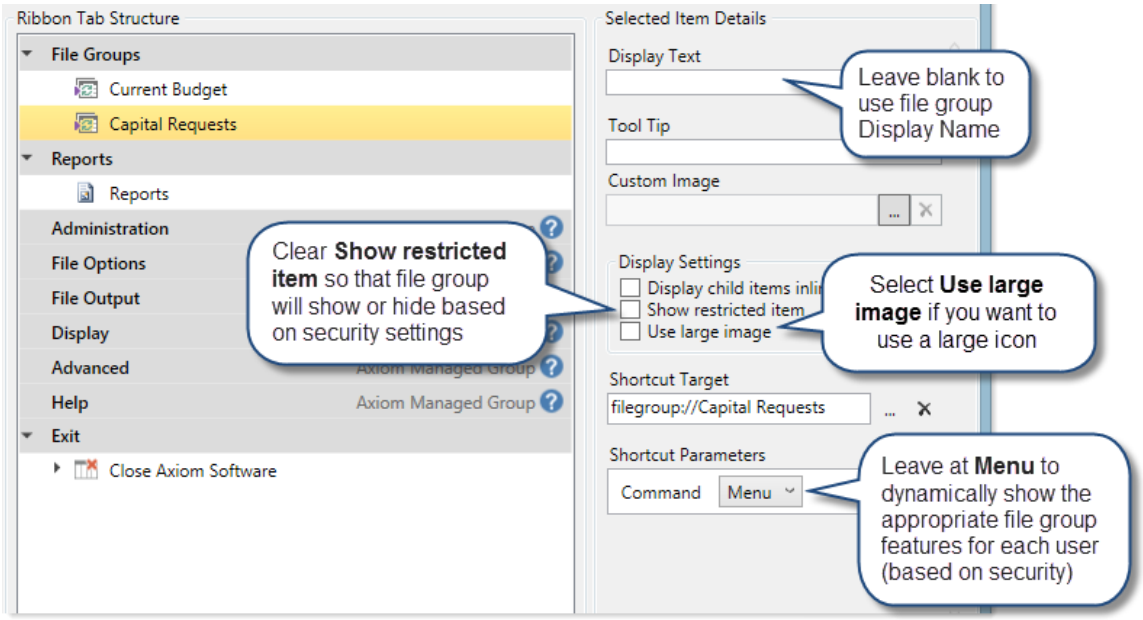
   Alternatively, you can add a new child item to the File Groups section, and then use the Shortcut Target field to browse to the desired file group or file group alias.

   If you place a file group alias on the ribbon tab, the currently assigned file group for that alias will display on the ribbon. If the alias is later edited to point to a different file group, then that newly assigned file group will now display on the ribbon.

   **NOTE:** If you are using file group categories, you can also choose to place a category on the ribbon tab. See the following section for more information.

5. Configure the settings for the new item on the ribbon tab as desired. Note the following:
   - By default the file group will display on the ribbon using the Display Name defined in the file group properties. You should leave the **Display Text** for the ribbon tab item blank unless you do not want to use the display name for the file group.

- Clear the **Show restricted item** check box so that the file group will only display if the user has security permission to access it. If you leave the option checked, then the file group will always display, but it will be grayed out if the user does not have permission to access it.
- In the Shortcut Parameters, leave the **Command** option set to **Menu**. This means that the item will dynamically display only the features that the user has rights to. If the user only has rights to open plan files, then the item will be a button that opens Open Plan Files. If the user has permissions to other file group features, then the item will be a drop-down list with the appropriate features.
- You can optionally select **Use large image** if you want the file group to display using a large-size icon. Typically, you would select or clear this option for all items in the File Groups group, so that all file groups display using the same size icon.




6. If you want to remove a file group from the ribbon tab, select it in the ribbon structure and then click **Delete**.
7. Click **Apply** or **OK** to save.

Any changes made to the ribbon tab will be reflected the next time you log in. The contents of the ribbon tab are determined at startup and do not reflect any subsequent changes made to the tab structure.

If you are using multiple versions of the AxiomMain ribbon tab, or if you are using other custom ribbon tabs and task panes, then you will need to update each ribbon tab where you want to display the file group.

### Using file group categories on the ribbon tab

If desired, you can display a file group category on a ribbon tab. If you place a category on the ribbon, then all file groups in that category will be included on the ribbon, in a drop-down menu underneath the category name.

In order to place a category on the ribbon, the **File Groups** section of Axiom Explorer must be displayed in category view. This is true whether you want to drag and drop a category from the Axiom Explorer pane in the ribbon tab editor, or whether you are using the Browse function of the Shortcut Target to browse the full Axiom Explorer dialog. To enable category view, right-click the **File Groups** header in Axiom Explorer, and then click **View > By Category**. The File Groups section now displays file groups organized by category. If a file group is not assigned to a category, it displays under **(No Category)**.

Keep in mind the following regarding the settings for the item when using categories:

- Categories do not have separate display names. The item will display on the ribbon using the category name. If you want to use a different name, you can define the **Display Text** for the item or change the name of the category.
- Again it is recommended to clear the **Show restricted item** check box. This will cause the file groups in the category to show or hide based on whether the user has access to each file group (including hiding the category entirely if the user has no access to any file groups in the category). If instead you leave **Show restricted item** checked, then all file groups in the category will display on the ribbon, and if the user does not have access to a file group then it will be grayed out.
- Categories do not have shortcut parameters. All file groups in the category will display using the "default" mode that dynamically displays the appropriate features for each user.

Keep in mind that if a category is placed on the ribbon, then all file groups in that category are eligible to be displayed on the ribbon (dependent on the user's security permissions and on the **Show restricted item** option). It is not possible to assign a file group to a category yet exclude that file group from the ribbon. You must remove the file group from the category if you do not want it to display on the ribbon.

By default, the file groups in a category will display in the order they were added to the category. If you want to define a different order, you can use the **Organize File Group Category** dialog to change the order. For more information, see Using file group categories.

# Using a Show On List column

You can define a Show On List column for a file group, to specify whether a particular plan code is included in file group processes.

This column is used for slightly different purposes depending on the "type" of file group.

- If the file group uses a predefined list of plan codes, then the primary use of this column is to prevent a plan file from being created for a particular plan code. For example, you may be mapping several plan codes to a single "parent" plan code, so that all plan development takes place in one plan file for those codes. In this case you do not want to create separate plan files for the mapped plan codes, so you remove them from file group processes using the Show On List column.
- If the file group adds plan codes "on demand," then the primary use of this column is to hide plan files that have already been created but no longer need to be included in file group processes. When a user creates an on-demand plan file, the entry in the Show On List column (if it exists) is automatically set to True so that the new plan file can be easily accessed and included in file group processes. However, if the plan file was created by mistake, or is "completed" and no longer needs to be available to file group processes, you can use the Show On List column to hide the plan file from file group dialogs (as an alternative to deleting the plan code and its associated plan file).

The Show On List column is not automatically created for a plan code table—if you want to use a Show On List column, it must be manually created. The Show On List column can be named anything you like, but it must be a Boolean column (True/False).

If Show On List is True for a plan code, then that plan code is available to file group processes. If Show On List is False for a plan code, then that plan code is not available to file group processes and does not display in file group dialogs such as Open Plan Files, Create Plan Files, etc. If no Show On List column is defined, then all plan codes are available to file group processes.

**NOTE:** If Show On List is False for a plan code, but the plan code already has an existing plan file, that plan file will still display in Axiom Explorer for administrator access and can be opened using the GetDocument function.

A plan code table can be used by multiple file groups, and therefore can have multiple Show On List columns. However, if the same settings apply to multiple file groups, the same Show On List column can be used.

# Using file attachments with file groups

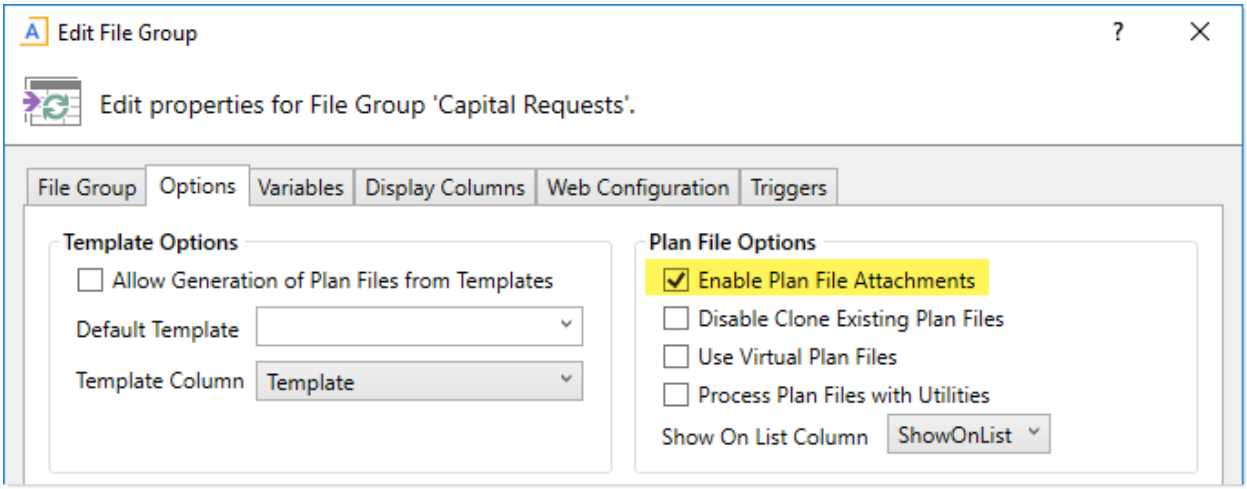
Axiom supports the ability to store file attachments that are associated with particular plan files. For example, you may be using a file group for capital planning, and as part of this process you want to store supporting documents for each capital request. Users can upload file attachments to be associated with a particular plan file, and then other users with access to the plan file can view these attachments.

You can store any type of file as a file attachment—for example, regular Excel files, Word files, PDF files, various types of image files, etc. Whether a user can open a particular type of file attachment depends on

whether the user has a program that can read that file format installed on their machine. If the file is Excel-compatible, then the attachment is opened in the current Axiom session; otherwise it is opened using the appropriate program on the user's computer, if present.

## Enabling file attachments for a file group

File attachment features are only available if they have been enabled for a particular file group. To enable file group attachments for a file group, select the **Enable Plan File Attachments** check box in the file group properties. This setting is located on the **Options** tab of the **Edit File Group** dialog.




If this check box is selected, then the **File Attachments** button is available on the ribbon while in plan files for that file group, so that users can view and manage attachments.

If you later clear this check box after attachments have been saved, those attachments will not be deleted, however the File Attachments button will no longer be present while in plan files for that file group. The GetPlanFileAttachment function will continue to work.

## Security for file attachments

Whether a user can view or manage attachments depends on that user's rights to the plan file, as determined by Axiom security (and potentially elevated by a plan file process).

- If a user has read-only rights to a plan file, then the user can view attachments for that plan file.
- If a user has read/write rights to a plan file, then the user can add, edit, and delete attachments for that plan file.

Administrators can always view and manage attachments for any plan file.

File attachments are stored in the Plan File Attachments folder within the file group folder in the Axiom file system. Each plan file has its own attachment sub-folder that is named using the plan file code. For example, attachments for plan file 21000 in the Budget 2022 file group would be stored as follows:

```
\Axiom\File Groups\Budget 2022\Plan File Attachments\21000
```

Administrators can access this area using Axiom Explorer. If needed, administrators can import and delete attachment files directly, and delete or create attachment folders. Non-administrator users cannot access these folders directly. They can only manage attachment files by using the management features in the software.

## Viewing attachments for a plan file

Users can view attachments for spreadsheet plan files in either of the following ways:

- While they are in a plan file, users can click the **File Attachments** button to view and open the attachments associated with that plan file, using the **Browse Attachments** dialog.
- The GetPlanFileAttachment function can be used in any Axiom file to open a particular plan file attachment, or to open the **Browse Attachments** dialog for a specified plan file.

For form-enabled plan files, users can view attachments by using the File Attachments panel in the task bar. Other form-specific features are also available. For more information, see the *Axiom Forms and Dashboards Guide*.

## Managing attachments for a plan file

Users with the appropriate rights can manage attachments for a spreadsheet plan file in either of the following ways:

- While they are in a plan file, users can click the **File Attachments** button to add, edit, or delete attachments for that plan file, using the **Manage Attachments** dialog.
- The ManagePlanFileAttachments function can be used in any Axiom file to open the **Browse Attachments** dialog for a specified plan file.

For form-enabled plan files, users can manage attachments by using the File Attachments panel in the task bar. Other form-specific features are also available. For more information, see the *Axiom Forms and Dashboards Guide*.

By default, a file attachment can be no larger than 10MB in size. Upload of the attachment is prevented if the attachment exceeds this size. This size limit is controlled by the **MaxFileAttachmentSizeKB** system configuration setting.

## Reporting on file attachments

The Axiom.PlanFileAttachments system table is available to bring a list of file attachments into an Axiom file. For example, you could create an Axiom query that brings in a list of all file attachments for a particular file group or for a particular plan file. The Axiom query could include a GetPlanFileAttachment function (or for a form-enabled plan file, a GetResourceLinkTag function) that automatically generates a link to the attachment.

# Using virtual plan files

File groups can optionally be configured to use *virtual plan files*. A virtual plan file means that the plan file is not persisted in the database. Instead, the plan file is dynamically created from template when it is accessed, and then when the plan file is closed this temporary file is discarded.

In order to use virtual plan files, the templates for your file group must be designed to support "rebuildable" plan files. This means that all data necessary for the operation of the file must be able to be queried into the file from the database. As users make inputs and change the data, any data that needs to be retained for future use must be saved to the database, so it can be queried back into the file the next time it is accessed.

Using virtual plan files can greatly improve system performance if the file group has many plan files. Because the physical plan files are not stored in the database, file access activities can be performed much faster. Additionally, using virtual plan files makes it easier to adopt ongoing changes to templates and calc methods, because the latest template and calc methods will be used each time the plan file is accessed.

## Rebuildable plan files versus persisted plan files

By default, plan files in Axiom are persisted files. When you create plan files from templates, each individual plan file is stored as a separate entity. When a user opens a plan file and makes edits, the plan file itself is saved as well as the planning data. This persisted file means that you do not need to save *all* of the user's inputs to the database; instead you only need to save the resulting planning data to be used for reporting. Any other inputs made by the user are saved in the plan file itself.

In contrast, a rebuildable plan file is specially designed to be re-created from the template each time the plan file is accessed. No data is stored in the plan file itself. Instead, the plan file works as follows:

- Data is queried from the database into the plan file.
- The user can make inputs and edits as needed within the plan file.
- All of the user's edits are saved from the plan file to the database.
- The next time a user opens the plan file, the saved data is queried back in from the database, so that the plan file starts at the same point the last user left it.

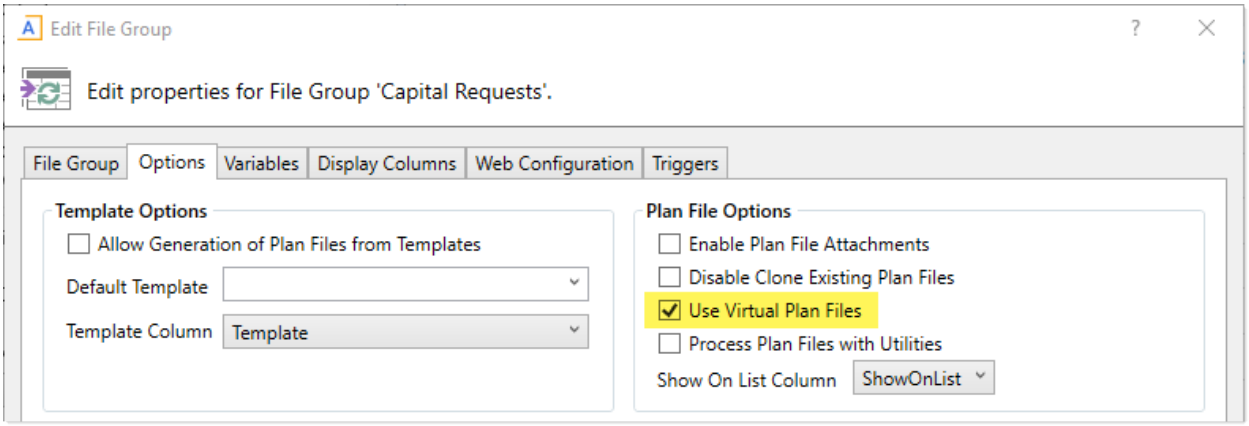
When using this rebuildable design, the plan file is not persisted as a permanent entity. Instead, the plan file serves as a temporary input form, as a means for the user to input data to be saved to the database. This means that any data that needs to be retained for future use must be saved to the database so it can be retrieved later, because the data will not be persisted in the file.

Form-enabled plan files are essentially rebuildable plan files, whether they are virtual or persisted. Because only data is saved in the Axiom forms environment, not the source file, form-enabled plan files must be designed as rebuildable. Even though the file may be persisted, the planning data in the file is "rebuilt" each time a user opens the form. Therefore, form-enabled files are ideal candidates for virtual plan files. If you have existing form-enabled plan files that you want to switch to virtual plan files, you may be able to make the change with little to no additional design effort.

Typically, traditional spreadsheet plan files are designed as persistent plan files. Although spreadsheet plan files can be designed as rebuildable and therefore made virtual, this requires a special design effort. Simply switching your existing persisted plan files to virtual plan files will not work, unless you know that these plan files are already designed as rebuildable (a rare situation).

## Enabling virtual plan files

If the templates in a file group are designed to support rebuildable plan files, it is recommended to enable virtual plan files to improve system performance. To enable virtual plan files, select the option **Use Virtual Plan Files**. This option is located on the **Options** tab of the **Edit File Group** dialog, in the **Plan File Options** section.




If this option is enabled, then individual plan files are not stored for the file group. You will still see the document records for the plan files in Axiom Explorer, but these records are effectively placeholders to store the necessary document properties and to allow navigation to the files. The actual spreadsheet plan files do not exist and are not retained in the database.

When a user opens a plan file using any means—such as from Open Plan Files, or from the Process task pane, or from a hyperlink in a form home page—Axiom looks up the template that was used to initially "create" the plan file, and generates a temporary copy of the plan file using the current copy of that template. The user can then work in this file as normal. Any save performed by the user only triggers a save-to-database, it does not trigger a file save. When the user closes the plan file, the temporary copy is discarded.

When the temporary copy is created, it behaves as a normal plan file and the Control Sheet settings are applied as normal. Axiom does not perform any special processing on the file. For example, if you want an Axiom query to run when the file is opened, then that query must be configured to run on open.

One advantage of virtual plan files is that design changes to templates and calc methods can be propagated to plan files immediately. The next time a user opens a plan file, or the next time Process Plan Files is performed, the latest versions of the template and calc methods are automatically used to create the virtual plan file.

Note the following feature considerations and limitations when using virtual plan files in a file group:

- Plan file restore points do not apply, because the physical plan files do not exist. It is not possible to restore to an earlier version of the plan file (though you could restore an earlier version of the template using document history, if appropriate).
- The options to save plan files and create restore points in Process Plan Files do not apply. No errors will occur if the options are enabled, but the options will have no effect.
- If you clone a file group with virtual plan files, and you choose to clone the plan files, then the placeholder document records will be copied to the new file group, so that the plan files in the new file group can be accessed.
- In plan file processes, if you enable the option to validate plan files before the task can be completed, the validation will only perform a save-to-database, not a file save. Also, if the option to prompt the user to complete the current process task after saving is enabled, this will still occur when the save-to-database is performed.
- Even though the plan file itself cannot be saved, users must still have read/write access to the plan file in order to upload and edit attachments, and to enable file locking for the virtual plan file. Plan file processes still operate as normal to elevate access to the plan files—so even though the user cannot save the file, the user's access will still be elevated to the equivalent of read/write.
- Once a virtual plan file is created, it will always have the same values for Last Modified Date and Last Modified By. Users performing save-to-database in the plan files do not affect these values because the file itself is never saved.

### Templates and virtual plan files

Virtual plan files must still be "created" from a template before they can be accessed. The creation process happens as normal, using Create Plan Files for standard file groups, and using the "create new" feature for on-demand file groups. When using virtual plan files, the creation process simply creates the placeholder document record, which allows users and system processes (like Process Plan Files) to access the virtual plan file.

When the virtual plan file is created, the document record is associated with the template that was used for the original creation. Whenever the plan file is accessed, Axiom uses the current version of that template to generate the temporary copy of the plan file. Any edits made to the template (and to calc methods) will be immediately available in the plan file. Therefore, you should use caution when editing a template that was used to create active plan files. Ideally, you should make changes in a test copy (leaving the current template intact for active plan files), then export the test copy and import it over the original template (to overwrite it and retain the document ID).

If you want a virtual plan file to use a different template, you must change the template assignment and then "re-create" the plan file using the Create Plan Files utility. This will overwrite the current placeholder document record with a new record that is associated with the new template.

### File locking for virtual spreadsheet plan files

When using virtual plan files, the plan file itself is never saved. The file is generated on-the-fly based on the template when it is needed, and is not persisted otherwise. Only data can be saved from the virtual plan file.

This behavior means that "read-only" as it refers to the file itself is meaningless, and the only distinction that matters is whether the user can save data. Therefore, for virtual spreadsheet plan files, the ability to save data determines whether the file is flagged as read-only, and determines whether file locking applies.

When a user opens a virtual spreadsheet plan file, the file is flagged as read-only ("R/O" on the file tab) if the user is unable to save data, for any reason. This is meant as a signal to the user that they will be unable to save any data changes. If the user is able to save data, then the plan file is locked to the current user, in order to prevent other users from saving data in the file at the same time.

In order for this file locking behavior to apply, the user must have **Read/Write** with **Allow Save Data** permissions to the plan file. When process management elevates user permissions, it automatically grants this level of access, so this behavior automatically applies to step owners in a plan file process. However, if you want some users to have edit permissions at all times (not just when they are the step owner), then it is recommended to grant those users Read/Write with Allow Save Data instead of just Read-Only with Allow Save Data. If the user's access level is Read-Only, then the file will not be locked.

**NOTE:** This behavior does not apply to virtual form-enabled plan files. File locking does not apply to Axiom forms, and Axiom forms are not flagged as read-only. If you want to control data saves in form-enabled plan files, the save locking feature for Axiom forms can be used.

# Configuring display columns for file group dialogs

Many file group features display lists of plan files in a grid, such as Open Plan Files, Process Plan Files, and Create Plan Files. You can configure display settings for these dialogs on a per file group basis to determine the following:

- Which columns are included in the list, and certain display attributes about those columns
- Which columns are used to sort the list
- Whether the list is a flat list of plan files, or grouped in expandable/collapsible categories

These settings are defined in the file group properties, on the **Display Columns** tab > **Plan File Columns** sub-tab. Only administrators and users with one of the following security permissions can edit file group properties: **Administer File Groups** and **Modify File Group**.

**To access the display column settings:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. Navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

**TIP:** You can also do this from the file group node in the Explorer task pane.

3. In the **Edit File Group** dialog, select the **Display Columns** tab, then select the **Plan File Columns** sub-tab.

**NOTE:** The settings on this tab do not affect the lists of plan files that are displayed in dialogs for plan file processes. Process columns can be customized on the separate **Process Columns** sub-tab.

## Defining the display columns

You can configure the columns to display in lists of plan files by default, as well as certain attributes about those columns. You can specify which columns are searchable, define alternate header text, and define the column width.

The current display columns are listed in the **Display Columns** box. By default, the following columns are selected to display:

- The key column of the plan code table
- Designated description columns for the plan code table
- Locked By
- Last Modified By
- Last Modified

You can include any column from the plan code table, from a data table that directly looks up to the plan code table, or from a reference table that the plan code table looks up to. You can also include various system columns relating to the modified status of the plan file.

Certain dialogs may automatically include other columns to display information relevant to that particular process. For example, the Create Plan Files dialog always displays the assigned template for each plan file and whether or not the file currently exists.

To configure which columns are included and in what order, click **Select Columns**. In the **Select Columns** dialog:

- To add a column, select the column in the left-hand pane of the dialog and then click **Add** to move it to the **Selected Columns** box.
- To remove a column, select the column in the **Selected Columns** box and then click **Remove**.
- To change the order of a column, select the column in the **Selected Columns** box and then click **Up** or **Down** to move it to the desired location.

Display attributes for each column are configured after the column has been added to the **Display Columns** box. To configure the display attributes for a column, select the column in the list and then click **Edit Column**. You can edit the following display properties:

| Item | Description |
|---|---|
| Header | The header text for the column. By default, this is the column name. You can customize this text if desired.<br><br>If the column is not on the plan code table, the fully qualified name is used by default. For example, if the plan code table is Dept, then if you add the `Dept.Region` column the default header value is just Region. But if you add the `CapData.Total` column, then the default header value is CapData.Total. |
| Width | The width of the column. By default, the display columns attempt to auto-size to a reasonable width for the column contents. If desired, you can enter a different width in pixels, up to a maximum of 500.<br><br>If you want to go back to using the default width, you can clear this field. |
| Searchable | Specifies whether the column is searchable, for dialogs that provide a search box (such as the Open Plan Files dialog). Select this check box if you want the contents of this column to be included in the search.<br><br>If no columns are flagged as searchable, then the search uses the key column and the designated description columns by default. Additionally, if the first display column is not the key column, it is also included in the search by default. |
| Custom Formatting | If the column values are numeric—meaning column data types of Integer (all types) or Numeric—then you can optionally define a custom display format for the values.<br><br>To define a display format, enter a valid Excel formatting string. These strings can be obtained as follows:<br>• Format a cell in a spreadsheet to use the desired display format.<br>• In the **Format Cells** dialog, on the **Number** tab, select the **Custom** category and copy the string in the **Type** box.<br><br>For example, this is the formatting string for a Currency format that shows the negative numbers in parentheses: `$#,##0.000_);($#,##0.000)`<br><br>Colors (such as red font for negative numbers) are not supported. Additionally, text replacement strings are only supported for zero values. Other advanced or unusual formats may not display as expected, so be sure to verify the column display.<br><br>If you do not define a custom display format, then the default formatting for the column's specified numeric type will be used. |

## Defining the sort columns

You can configure the default sort order for lists of plan files. The sort order is specified based on one or more columns in the plan code table. If multiple columns are selected, then the primary sort is the top column, the secondary sort is the next column, and so on.

The current sort columns are listed in the **Sorting Columns** box. By default, the key column of the plan code table is selected as the sort column. For example, if the plan code table is DEPT, then the plan files will be sorted in order of department codes, in ascending order.

To define the sort columns, click **Select Columns**. In the **Select Columns** dialog:

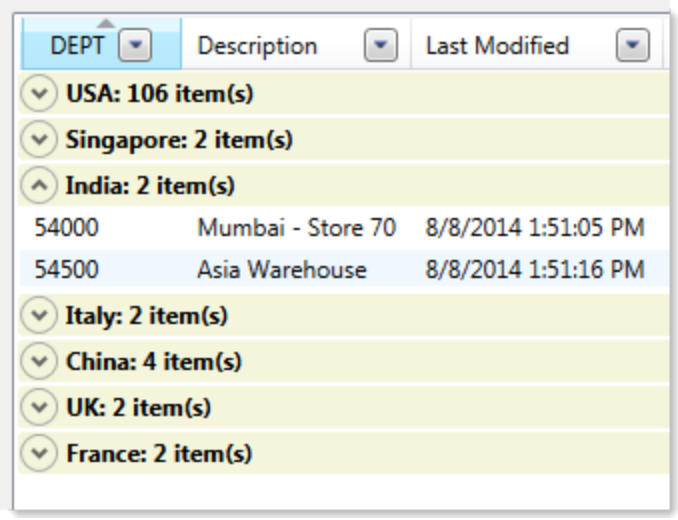
- To add a column, select the column in the left-hand pane of the dialog and then click **Add** to move it to the **Selected Columns** box.
- To remove a column, select the column in the **Selected Columns** box and then click **Remove**.
- To change the order of a column, select the column in the **Selected Columns** box and then click **Up** or **Down** to move it to the desired location.

By default, the sort is ascending order. Once a column has been added to the **Sorting columns** box, you can change the order by using the drop-down list next to the column name.

## Defining a grouping column

You can optionally specify a grouping for grids that display lists of plan files. By default, no grouping is applied. This means that plan files are displayed in a flat list (one row per plan file).

Alternatively, you can specify a grouping column such as Region or Company, and then plan files will be displayed in expandable / collapsible groupings. For example, the following screenshot shows a list grouped by Country:




Remember that most end users may only have access to one plan file or a small handful of plan files, in which case grouping may be unnecessary. Administrators and other power users with access to many plan files can always apply a grouping to the list manually when they are in a dialog. End users can do this as well, but in most cases they do not need to.

The current grouping column, if any, is listed in the **Grouping column** box.

- To define the grouping column, click **Select**. In the **Column Chooser** dialog, select the column to group by. You can select any column in the plan code table or a lookup reference table.
- To clear the grouping column, click **Clear**.

3

# File Group Variables

Each file group can have a set of defined variables to be referenced in file group components such as templates, drivers, and utilities. These variables can be used for any purpose, but it is recommended that they be used for some of the key properties of the file group, such as:

- The target tables for saving data, including driver tables
- The source tables for actuals and other supporting data
- The picklist tables used by the file group
- Certain key assumptions to be used for scenario planning

The purpose of file group variables is to avoid "hard-coding" these values within file group files, so that when the file group is cloned these values are not also inadvertently copied. Variables are also used to enable rapid scenario planning for a file group. Basically, if there is an essential property that you would want to change when cloning a file group or creating a scenario, that property should be set up as a file group variable.

For example, imagine that you define the target table for saving plan data within a driver file, so that the template and the resulting plan files reference this value. You then clone the file group to prepare for another year of planning. If you forget to update this value in the drivers of the new file group, then that file group will start saving data to the wrong table. On the other hand if you define this value as a file group variable, then when you clone the file group part of the cloning process will be to define new values for the file group variables, so the file group will start out with the correct values.

Once the variables and their values have been defined, you can reference them in file group components using the GetFileGroupVariable function. For example, if you have defined a variable named PlanTable that defines the target table for saving planning data, you can reference that value as follows:

```
=GetFileGroupVariable("PlanTable")
```

This would return BGT2022 if that is the current value defined for the variable in the file group settings.

File group variables can also be referenced in report files that are not associated with a file group, but in that case you must specify both the file group name and the variable name in the GetFileGroupVariable function (because there is no "current file group" for the report). File group aliases can be used in this case so that the function can dynamically point to different file groups as needed.

# Using general variables for file groups

You can use general variables in file groups to:

- Define certain values globally for the file group, such as certain key planning assumptions.
- Define general values to be used in other file group settings and variables.

General variable values can be returned in files by using the GetFileGroupVariable function as normal. General variables are also often used in other file group settings. For example, you can define a general variable for `{ShortFGName}` and then reference that value in settings such as the Tab Prefix and in other variables.

All general variable creation and configuration takes place on the **General Variables** sub-tab in the file group properties. To access this area:

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. In the **Axiom Explorer** dialog, navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

   **TIP:** You can also access this dialog by right-clicking a file group in the Explorer task pane.

3. Select the **Variables** tab, then select the **General Variables** sub-tab.

From this area, you can create new variables, edit existing variables, and delete variables. Only administrators or users with the **Administer File Group** permission can access this area.

## Creating general variables

You can create new general variables as needed.

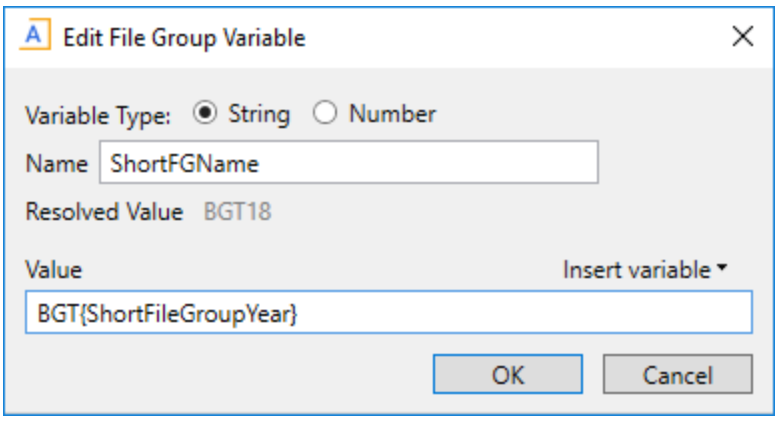
**To create a general variable:**

1. On the **General Variables** sub-tab, click **New**.
2. In the **Edit File Group Variable** dialog, complete the following fields:

| Item | Description |
|---|---|
| Variable Type | The data type of the variable, **String** or **Number**. The default setting is String. You should select Number if the variable stores a numeric rate that will be used in spreadsheet calculations. |
| Variable Name | The name of the variable. The name should be brief, yet self-explanatory. For example, `SalaryEscalator` to define a salary escalator value to be used in scenario planning. |
| Variable Value | The value of the variable. You can type in a value, and/or you can reference another file group variable to define the value. |

| Item | Description |
| --- | --- |
| | To reference another file group variable within the value, enter the variable name in curly brackets. You can also use the **Insert variable** tool to insert any user-defined general variable within the file group, or to insert a built-in variable such as `{FileGroupYear}`.<br><br>For example, if the variable name is SalaryEscalator, then you might type in `.03` as the value. |

Once the variable value has been defined, the **Resolved Value** shows the resolved value of the variable. This is helpful if the variable value references other variables, to see what the value ultimately resolves to.




*Example general variable*

3. Click **OK** to close the Edit File Group Variable dialog and add the new variable to the Variables grid.
4. Click **Apply** or **OK** in the Edit File Group dialog to save your changes.

## Managing existing general variables

To edit a general variable, select the variable in the grid and then click **Edit**. Generally speaking, you should not change a variable name once any file group components have been created that reference the name, such as plan file templates, drivers, or utilities. If you change the name, you will need to manually find and update any GetFileGroupVariable references to the old name in file group components.

To delete a general variable, select the variable in the grid and then click **Delete**. Before deleting a file group variable, you should be absolutely sure that the variable is not needed. Deleting an in-use variable will cause GetFileGroupVariable functions that reference the variable to return an error.

# Using table variables for file groups

You can use table variables in file groups to:

- Identify and reference the tables associated with the file group.
- Control which tables the file group can save data to.
- Automatically create new tables as needed when cloning the file group or creating a file group scenario.

Table variable values can be returned in files by using the GetFileGroupVariable function as normal. Using this approach, you can dynamically return the names of the current tables associated with the file group, and use those tables as needed. By using variables to define the tables, you can control the data queries and data saves for the file group from a single location. Any changes made to the table variables will flow through all files in the file group that reference the variables.

All table variable creation and configuration takes place on the **Table Variables** sub-tab in the file group properties. To access this area:

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. In the **Axiom Explorer** dialog, navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

   **TIP:** You can also access this dialog by right-clicking a file group in the Explorer task pane.

3. Select the **Variables** tab, then select the **Table Variables** sub-tab.

From this area, you can create new variables, edit existing variables, and delete variables. Only administrators or users with the **Administer File Group** permission can access this area.

## About table variables

Any table that you plan to query data from or save data to—from within any file that belongs to the file group—should be defined as a table variable and then referenced in file group components using the GetFileGroupVariable function. This includes tables such as the following:

- Data tables that hold GL actuals and other historical data, including planning data from prior years. Plan files query data from these tables in order to build out the starting point for the plan, but they do not save data to these tables.
- Data tables to store the planning data that is calculated in plan files. Plan files save data to these tables.
- Reference tables or document reference tables that store the driver data used by the plan files. Driver files save data to these tables.

**NOTES:**

- It is not typically necessary to define query-only reference tables such as DEPT and ACCT as table variables, although you can if desired. Reference tables usually remain constant, and in that case, there is no advantage to using a table variable versus simply referencing the table directly. However, if the file group needs to save data to these tables, then they should be defined as table variables.
- If you are using picklist tables with a file group, those tables should be defined as picklist variables instead of table variables, so that you can use the additional picklist-specific features. However, if necessary, you can create a table variable that resolves to a picklist table—for example, if you need to save data to the picklist table from the file group.

The reason for using table variables is so that you can dynamically change which tables are used by the file group when cloning the file group or when creating file group scenarios. If instead the table names are "hard-coded" into the template or the drivers, then you may forget to update these names or accidentally overlook a reference to a particular table. This can result in querying data from the wrong tables, or worse, saving data to the wrong tables. Using table variables means there is one place where you manage all of your table names for the file group, and you can update these names as part of the cloning process or scenario creation.

Using table variables also provides the following benefits:

- When cloning a file group, if a table variable resolves to a table that does not yet exist, the cloning process will create the table for you. For example, if the variable resolved to BGT2021 in the old file group and resolves to BGT2022 in the new file group, the cloning process will create BGT2022 by cloning the structure of BGT2021.
- When creating a file group scenario, the scenario wizard will automatically create new data tables to hold the data for the scenario. This is accomplished by cloning the target tables for any table variables that have **Allow file group to save data to this table** enabled. For example, if the original file group saves data to BGT2022, the scenario wizard will automatically create a table such as BGT2022_V1, as long as BGT2022 is set up as a table variable with saves enabled. You can use the default scenario cloning behavior for these tables, or you can configure the table variable to override the default behavior as needed.
- You can limit the tables where the file group can save by using the optional setting **Restrict Saves To Tables Defined Below**. If this setting is enabled, then the file group can only save to tables that are defined as table variables with **Allow file group to save data to this table** enabled.

## Creating table variables

You can create new table variables as needed.

**To create a table variable:**

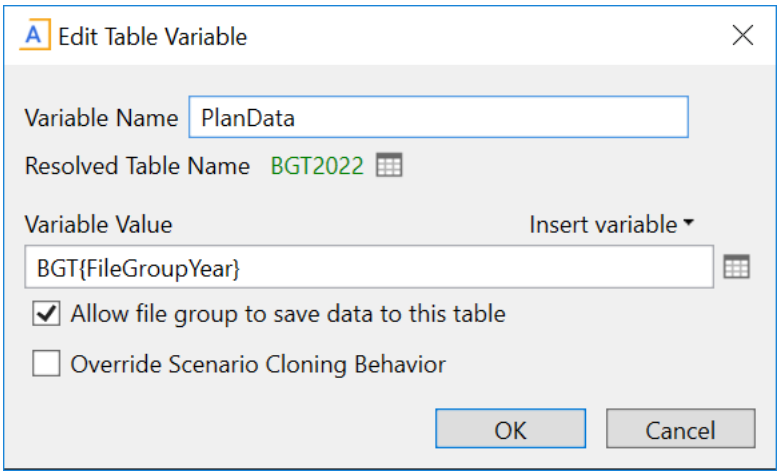
1. On the **Table Variables** sub-tab, click **New**.

2. In the **Edit Table Variable** dialog, complete the following fields:

| Item | Description |
|---|---|
| Variable Name | The name of the variable. The name should be brief, yet self-explanatory. For example, `PlanData` to define the name of the target table to save planning data. |
| Variable Value | The value of the variable. You can type in a value, and/or you can reference another file group variable to define the value.<br><br>To reference another file group variable within the value, enter the variable name in curly brackets. You can also use the **Insert variable** tool to insert any user-defined general variable within the file group, or to insert a built-in variable such as `{FileGroupYear}`.<br><br>Alternatively, you can use the table chooser to the right of the box to choose a table and set the variable value to that specific table name.<br><br>For example, if the variable name is PlanTable for the table where planning data will be saved, then the variable value might be `BGT {FileGroupYear}`. This would resolve to BGT2022 if the file group year is 2022. |
| Allow file group to save data to this table | Specifies whether the file group is intended to save data to the table referenced by this variable. You should clear this option if the file group will only ever query data from this table. By default, this option is selected.<br><br>This setting is used for two purposes:<br>• It determines which tables the file group is allowed to save data to. This restriction is only enforced if the setting **Restrict saves to tables defined below** is also enabled.<br>• It determines which tables will be cloned when creating a file group scenario. Only tables where the file group saves data will be cloned. |
| Override Scenario Cloning Behavior | Specifies whether the default cloning behavior is used for this table when creating a scenario. This option only applies if **Allow file group to save data to this table** is enabled.<br><br>By default this option is not selected, which means that the default cloning behavior is used for this table. For more information on the default table cloning behavior when creating a scenario, see Setting up table variables for scenario creation.<br><br>If you enable this option, then a new **Scenario Cloning Behavior** option becomes available for you to specify the cloning behavior for this table. |

| Item | Description |
|---|---|
| Scenario Cloning Behavior | This option is only available if **Override Scenario Cloning Behavior** is enabled. Select one of the following:<br>• **Ignore**: The table is not cloned when creating a scenario. The scenario will continue to use the same table as the original file group.<br>• **Structure**: The table is cloned when creating a scenario, but only the structure is copied, not the data. Essentially, the table is empty until data is saved to it from the scenario.<br>• **Structure and Data**: The table is fully cloned when creating a scenario—both structure and data. This means that the scenario starts with the same data as the original file group. |

Once the variable value has been defined, the **Resolved Table Name** shows the table name that the variable resolves to. If the resolved table name does not match an existing table, a warning icon is displayed. This may be expected if the target table has not yet been created.




*Example table variable*

3. Click **OK** to close the Edit Table Variable dialog and add the new variable to the Table Variables grid.
4. Click **Apply** or **OK** in the Edit File Group dialog to save your changes.

## Restricting data saves based on table variables

You can limit the tables that the file group is allowed to save data to by using the optional setting **Restrict saves to tables defined below**. This setting is located at the top of the **Table Variables** sub-tab, above the variable grid.

If this setting is enabled, then the file group can only save to tables that meet both of the following criteria:

- A table variable in the file group resolves to the table name.
- The table variable has **Allow file group to save data to this table** enabled.

If the target table for the save is not associated with a file group variable, then the save will be prevented. The save will also be prevented if the target table is associated with a file group variable, but **Allow file group to save data to this table** is not enabled.

This feature is a safety measure that is intended to help prevent accidentally saving to the wrong tables—for example, if someone "hard-coded" a table name into a template instead of referencing a variable. For example, if a template is set up to save to BGT2021 but BGT2021 is not defined as a table variable for the file group, then the save will be prevented.

### Managing existing table variables

To edit a table variable, select the variable in the grid and then click **Edit**. When editing a variable, keep in mind the following:

- Generally speaking, you should not change a variable name once any file group components have been created that reference the name, such as plan file templates, drivers, or utilities. If you change the name, you will need to manually find and update any GetFileGroupVariable references to the old name in file group components.
- If you change the variable value so that it resolves to a different table, be sure you understand the impacts of the change within the file group. Changing the variable value may impact processes such as the data being queried into plan files, and the tables being saved to. The ability to modify file group behavior by changing variable values is the purpose of file group variables, but you should take care to ensure you are setting the values as appropriate for the file group.

To delete a table variable, select the variable in the grid and then click **Delete**. Before deleting a file group variable, you should be absolutely sure that the variable is not needed. Deleting an in-use variable will cause GetFileGroupVariable functions that reference the variable to return an error.

# Using picklist variables for file groups

You can use picklist variables in file groups to:

- Identify and reference the picklist tables associated with the file group. Picklist tables are special reference tables that are used to define lists of values from which users can make selections.
- Define picklist variable properties that can be used to impact how the picklist is used in the file group.

Picklist variables are special kinds of table variables. Each picklist variable must resolve to a picklist table name.

Picklist variable values can be returned in files by using the GetFileGroupVariable function as normal. Using this approach, you can dynamically return the names of the current picklist tables for the file group, and use those tables as needed. Additionally, picklist variable properties can be returned in files by using the GetFileGroupVariableProperty function, such as to return whether the picklist variable is flagged as required. You can then configure the file to behave as needed depending on the variable property.

All picklist variable creation and configuration takes place on the **Picklist Variables** sub-tab in the file group properties. To access this area:

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. In the **Axiom Explorer** dialog, navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

   **TIP:** You can also access this dialog by right-clicking a file group in the Explorer task pane.

3. Select the **Variables** tab, then select the **Picklist Variables** sub-tab.

From this area, you can create new variables, edit existing variables, and delete variables. Only administrators or users with the **Administer File Group** permission can access this area.

**NOTE:** If **Restrict saves to tables defined below** is enabled on the **Table Variables** sub-tab, then it is not possible to save data to the target table of a picklist variable. In most cases, this behavior does not cause any issues because it is typically not necessary to save data to picklist tables from within a file group. However, if you do need to do this, then you must define a table variable that resolves to the picklist table, and configure that variable to allow saving data. This table variable can be used in addition to the picklist variable or instead of the picklist variable, depending on whether you need the advanced configuration options of picklist variables.

## Creating picklist variables

You can create new picklist variables as needed.

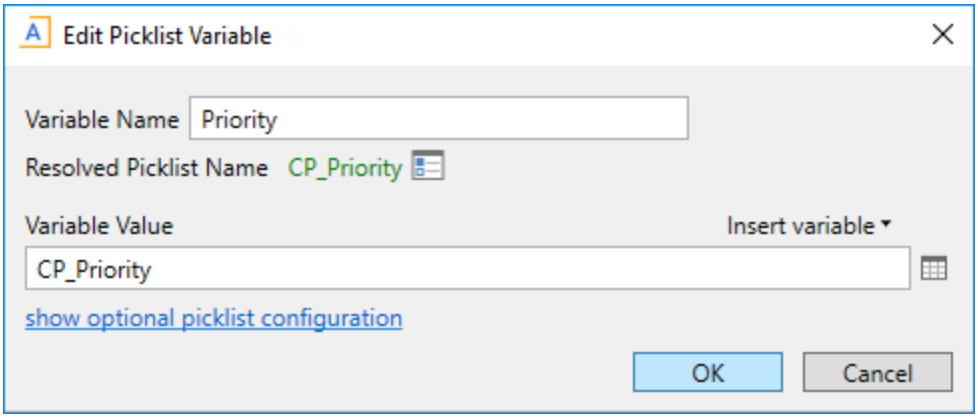
**To create a picklist variable:**

1. On the **Picklist Variables** sub-tab, click **New**.
2. In the **Edit Picklist Variable** dialog, complete the following fields:

| Item | Description |
|---|---|
| Variable Name | The name of the variable. The name should be brief, yet self-explanatory.<br><br>For example, `Category` to define the name of the target picklist table that holds the list of categories for users to select. |

| Item | Description |
|---|---|
| Variable Value | The value of the variable. You can type in a value, and/or you can reference another file group variable to define the value.<br><br>To reference another file group variable within the value, enter the variable name in curly brackets. You can also use the **Insert variable** tool to insert any user-defined general variable within the file group, or to insert a built-in variable such as `{FileGroupYear}`.<br><br>Alternatively, you can use the table chooser to the right of the box to choose a table and set the variable value to that specific table name.<br><br>For example, if the variable name is Category for the picklist table that holds capital planning categories, then the variable value might be `CP_Category` or `CP{FileGroupYear}_Category` (if each year of planning uses different categories). The latter example would resolve to CP2022_Category if the file group year is 2022. |

Once the variable value has been defined, the **Resolved Picklist Name** shows the picklist table name that the variable resolves to. If the resolved table name does not match an existing table, a warning icon is displayed. This may be expected if the target table has not yet been created.




*Example picklist variable*

3. If the variable value resolves to an existing picklist table, then you can optionally click **show optional picklist configuration** to expose the advanced picklist settings. In this section, you can configure the following:

   - **Picklist Column Configuration**: You can optionally associate this picklist variable with an existing column in the plan code table that has a lookup relationship to the picklist table. If the plan code table does not already have a column with a lookup relationship, you can create a new column. For more information, see Associating a picklist variable with a column in the plan code table.

- **Picklist Enablement**: If a **Picklist Enablement Column** has been defined for the file group, you can specify the values in that column for which this picklist variable should be enabled. For more information, see Dynamically enabling picklists.
- **Picklist Properties**: You can specify whether the picklist variable should be flagged as required. For more information, see Required variables.

4. Click **OK** to close the Edit Picklist Variable dialog and add the new variable to the Picklist Variables grid.
5. Click **Apply** or **OK** in the Edit File Group dialog to save your changes.

### Associating a picklist variable with a column in the plan code table

You can optionally associate a picklist variable with a validated column in the plan code table. This feature serves two purposes:

- You can use the GetFileGroupVariableProperty function to return the associated column, and then reference that column as needed in file group components such as plan file templates and utilities.
- Axiom can automatically create the validated column for you if it does not already exist on the table.

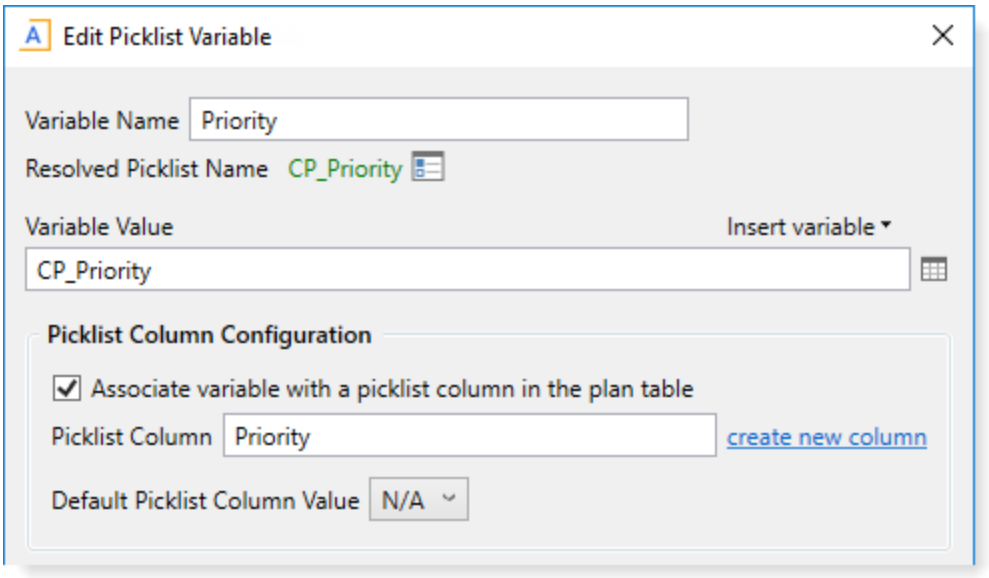
To associate a picklist variable with a column, use the **Picklist Column Configuration** section in the **Edit Picklist Variable** dialog. If this section does not already display in the dialog, click **show optional picklist configuration** to expose the advanced variable settings.

Complete the following properties to associate the variable with a column:

| Item | Description |
|---|---|
| Associate variable with a picklist column in the plan table | Select this check box to enable the ability to associate a column with the picklist variable.<br><br>If necessary, you can clear this check box to remove an existing association. Clearing the check box only affects the variable properties; it will not delete or modify the column in the plan code table.<br><br>An existing association may be removed automatically if the variable value changes (so that the variable now resolves to a different picklist table), or if the plan code table is modified so that the associated column is deleted or no longer looks up to the target picklist table. |

| Item | Description |
|---|---|
| Picklist Column | If the plan code table already contains a validated column that looks up to the target picklist table, that column is listed here by default. If the plan code table contains multiple eligible columns, the field becomes a drop-down list where you can select the desired column.<br><br>If the plan code table does not contain any eligible columns (or if you do not want to use the existing columns), click **create new column** to create a new column on the plan code table.<br>• The Picklist Column field is automatically populated with a column name using the format *PicklistTable*_Col. You can modify this name as needed.<br>• When you save the variable and exit the Edit File Group dialog, a new integer column will be created on the plan code table using this name. The new column has a lookup to the target picklist table. |
| Default Picklist Column Value | Specify the default value for the associated column in the plan code table. By default, this is set to the default value for the target picklist table. You can select any value in the picklist table if you want the default value for this column to be something different.<br><br>When you save, Axiom will set the default value in the associated column to the corresponding code for the selected value. For example, if the default picklist column value is left at N/A, then the default value of the associated column will be set to the corresponding code of 0. If new data is saved to the plan code table and no value is specified for the associated column, the default value is used. |

In the following example, the Priority picklist variable is associated with a column in the plan code table. That table has an existing column named Priority with a lookup to the CP_Priority picklist table, so that column is selected by default.




In the plan file template and other files, you can use the GetFileGroupVariableProperty function to return the associated column name. For example:

```
=GetFileGroupVariableProperty("Priority","ColumnName")
```

This function returns the name of the associated column, Priority. The function assumes the current file group if used in a file that belongs to a file group. When using the function in a report, you would have to specify the file group ID using the optional third parameter.

## Dynamically enabling picklists

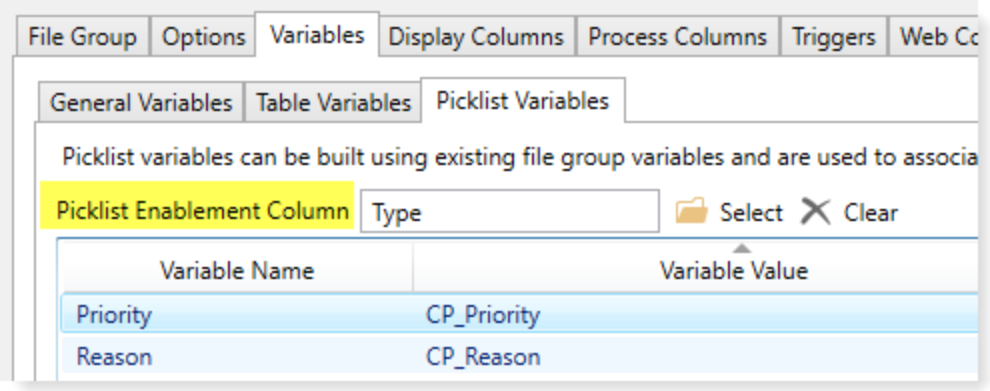
Your plan files may be designed so that certain picklists only apply to plan files that meet a certain criteria. You can use picklist variables to dynamically enable or disable certain picklists as follows:

- You can optionally specify a **Picklist Enablement Column** on the plan code table that holds values to determine whether certain picklists should be enabled or not for each plan file. For example, a column named Type that specifies the type of capital request.
- For each picklist variable, use the **Picklist Enablement** section to specify which values in the enablement column the picklist should be enabled for. Certain picklists may only apply to certain types.
- In the plan file template (or in other files as needed), you can use the GetFileGroupVariableEnablement function to return whether a picklist variable is enabled for a specified value in the picklist enablement column. You can then use formulas to dynamically show or hide the picklist. For example, you could hide a combo box that references the picklist if the current plan file is assigned to a type for which the picklist variable is not enabled.

### Picklist Enablement Column

The **Picklist Enablement Column** is specified at the top of the **Picklist Variables** sub-tab, and applies to all picklist variables. To specify a column, click **Select** to open the **Column Chooser** dialog. You can select any column on the plan code table that is a validated column with a lookup to a picklist table.

In this example, the column Type is the specified picklist enablement column. The values in this column can be used to dynamically enable or disable certain picklist variables per plan file.




### Picklist Enablement for each picklist variable

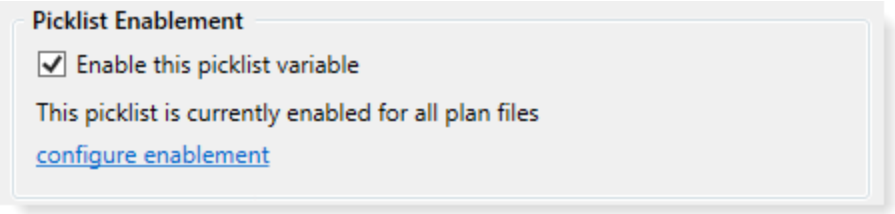
Once a picklist enablement column has been designated for the file group, you can specify which values in that column each picklist variable should be enabled for. By default, each picklist variable is enabled for all values in the picklist enablement column.

**NOTE:** Specifying enablement values for a variable simply updates the variable properties, so that you can return the enablement status using the GetFileGroupVariableEnablement function. Axiom does not provide any built-in functionality to enable or disable the variable within plan files. It is up to the file designer to return the enablement status and then build the desired solution as needed—for example, to hide a combo box that uses a certain picklist if the picklist variable is disabled.

To select the enablement values for a picklist variable, use the **Picklist Enablement** section in the **Edit File Group Variable** dialog. If this section does not already display in the dialog, click **show optional picklist configuration** to expose the advanced variable settings. This section is only present in the dialog if a picklist enablement column has been specified for the file group at the top of the Picklist Variables tab.

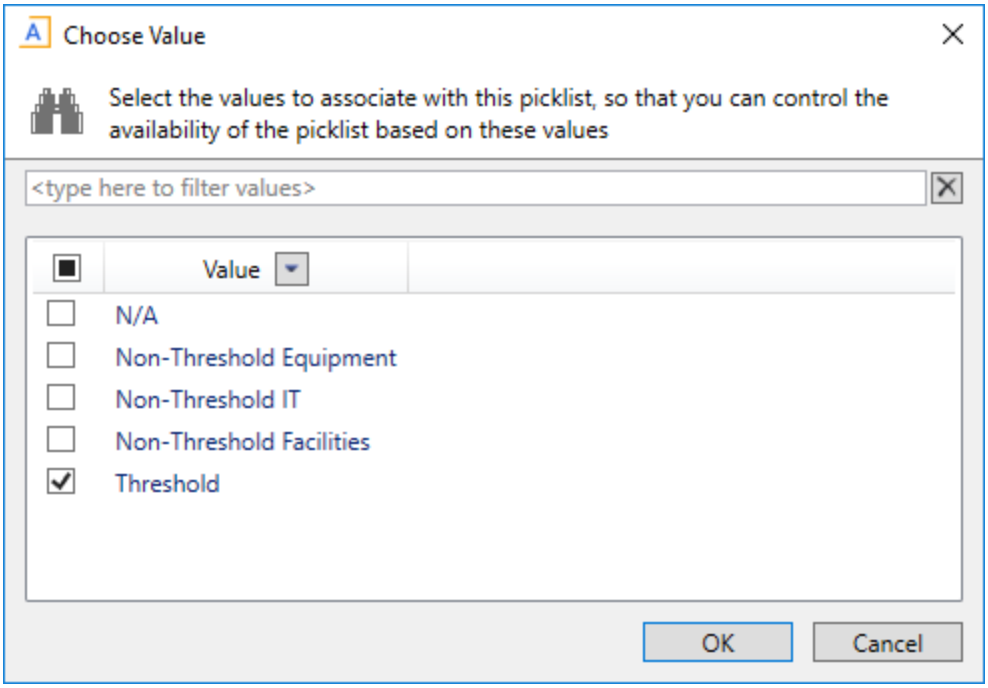
**To specify enablement values for the variable:**

1. Click **configure enablement**.



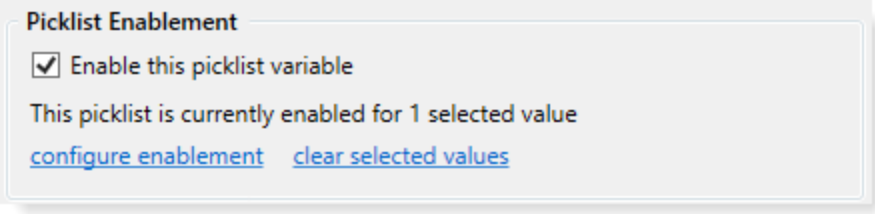

2. In the **Choose Value** dialog, select the values for which you want this variable to be enabled, and then click **OK**. This dialog displays all values in the picklist that the picklist enablement column looks up to.

   To continue the previous example, the picklist enablement column Type looks up to the Type picklist table. The Choose Value dialog shows the values in the Type picklist table.




**NOTE:** When you first access this dialog for a variable, no values are selected. This means that the variable is enabled for all values. Once you select one or more values and save the changes, the variable is now enabled for only those values.

When you return to the **Edit Picklist Variable** dialog, the Picklist Enablement section is updated to reflect the number of enabled values.




If you need to modify the enabled values later, you can do any of the following:

- Click **configure enablement** to open the Choose Value dialog again, and edit the selected values.
- Click **clear selected values** to clear all selected values, which means the variable is enabled for all values (the default behavior when no values are selected).
- Clear the **Enable this picklist variable** check box to make the variable disabled for all values.

### Returning the enablement status

In the plan file template and other files, you can use the GetFileGroupVariableEnablement function to return the enablement status of the variable for a given value. For example:

```
=GetFileGroupVariableEnablement("Priority",5)
```

This function returns True if picklist variable Priority is enabled for value 5, and False if it is not enabled for value 5. In most cases you would be returning the value in the picklist enablement column on a per plan file basis using a GetData data lookup or function, and then use a cell reference in the 2nd parameter.

> **NOTE:** Although the Choose Value dialog for picklist enablement shows the text values for the associated picklist, the corresponding integer value must be used in the GetFileGroupVariableEnablement function. The text values are not recognized by the function.

For example, if the plan file is form-enabled, then you may have a Select tag that points to the picklist table associated with the variable, so that the user can select a value from that picklist. If the function returns True, the row with the Select tag shows as normal. if the function returns False, you can dynamically hide the `[Row]` tag, so that the user is not prompted to select a value for this picklist.

The same basic principles apply to a spreadsheet plan file, except in this case you would probably set up dynamic view tags so that the applicable row or column associated with the picklist variable could be hidden in the spreadsheet.

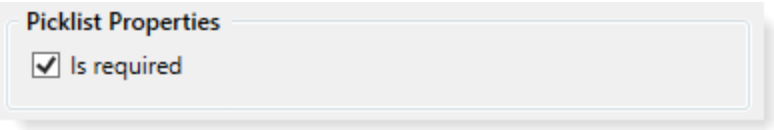
The GetFileGroupVariableProperty function can also be used to return the list of enabled values, instead of returning the enablement status of a specific value.

## Required variables

You can optionally flag a picklist variable as required, so that you can enforce selection from the target picklist table. For example, you might want to:

- Set up custom save validation so that the user cannot save data unless they have selected a non-default value from the picklist.
- Set up a picklist refresh variable so that it is a required variable, meaning the user must select a value in order to refresh data.

To flag a picklist variable as required, select the **Is required** check box in the **Picklist Properties** section of the **Edit Picklist Variable** dialog. If this section does not already display in the dialog, click **show optional picklist configuration** to expose the advanced variable settings.




**NOTE:** Enabling the **Is required** check box simply updates the variable properties, so that you can return the required status using the GetFileGroupVariableProperty function. Axiom does not provide any built-in functionality to require the picklist value when saving data to the database or performing other actions. It is up to the file designer to return the required status and then build the desired solution as needed—for example, to enable custom save validation if the picklist variable is flagged as required.

In the plan file template and other files, you can use the GetFileGroupVariableProperty function to return whether the variable is flagged as required. For example:

```
=GetFileGroupVariableProperty("Priority","IsRequired")
```

This function returns True if the Priority variable is flagged as required, and False if not. The function assumes the current file group if used in a file that belongs to a file group. When using the function in a report, you would have to specify the file group ID using the optional third parameter.

## Managing existing picklist variables

To edit a picklist variable, select the variable in the grid and then click **Edit**. When editing a variable, keep in mind the following:

- Generally speaking, you should not change a variable name once any file group components have been created that reference the name, such as plan file templates, drivers, or utilities. If you change the name, you will need to manually find and update any GetFileGroupVariable references to the old name in file group components.

- If you change the variable value so that it resolves to a different table, be sure you understand the impacts of the change within the file group. Changing the variable value may impact processes such as the data being queried into plan files, and the tables being saved to. It may also impact the associated picklist column, if set. The ability to modify file group behavior by changing variable values is the purpose of file group variables, but you should take care to ensure you are setting the values as appropriate for the file group.
- If you modify any of the advanced variable properties such as the associated column, picklist enablement, and required status, this may affect the behavior of the variable in existing file group components. Make sure you understand how the variable is being used before making any changes to the properties.

To delete a picklist variable, select the variable in the grid and then click **Delete**. Before deleting a file group variable, you should be absolutely sure that the variable is not needed. Deleting an in-use variable will cause GetFileGroupVariable functions that reference the variable to return an error.

### Editing product placeholder variables

Clients using installed product systems may have access to "placeholder" picklist variables that were installed by the package. These placeholder variables do not have a defined value and do not currently resolve to any picklist tables. It is intended that you can create custom picklist tables as needed, and then update these placeholder variables to reference your custom tables. File group components will then use your custom picklist tables.

When editing a placeholder variable to reference a custom picklist table, note the following:

- The advanced configuration section of the variable properties is automatically visible. It is not necessary to click the link to expose it.
- The **Picklist Column Configuration** section is not optional for placeholder variables. When you save the variable, Axiom will automatically create an associated column in the plan code table that looks up to the target picklist.

For more information on using picklist variables with your installed product, see the separate documentation provided for the product.

# Using years as variables in file groups

If a file group year is defined for the file group, you can use that value as a variable in the file group name and in other file group variables. You can also use built-in variables that automatically calculate a number of years before and after the file group year. Using this approach, you can dynamically update all associated years for a file group by changing one value.

## File group year as a variable

To use the defined file group year as a variable, use the following syntax:

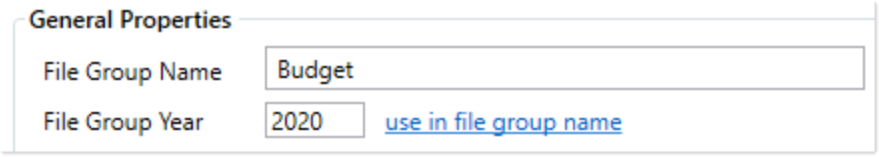
```
{FileGroupYear}
```

OR

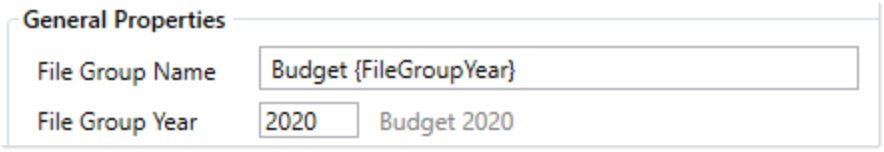
```
{ShortFileGroupYear}
```

The FileGroupYear variable returns the full year value (for example: 2022), and the ShortFileGroupYear variable returns the two-digit version of the year (for example: 22). The two-digit version is automatically calculated based on the value for the file group year.

Either of these variables can be used in the file group name, so that the current file group year is always reflected in the name (for example, "Budget 2022"). You can type the variable into the File Group Name field directly, or you can click the **use in file group name** link that displays to the right of the box after the file group year is defined. (If you want to use the ShortFileGroupYear in this field then you must manually type the variable.)




If the variable is present in the file group name, the resolved name will display in gray text to the right of the File Group Year box, as shown in the following screenshot:




The year variables can also be used in the file group display name and the tab prefix, as well as in other file group variables.

## Calculate years based on the file group year

File groups often query and save data from one or more years of data tables. For example if the purpose of the file group is to calculate budget data for the 2022 budget, then the file group may query data from GL2020 and GL2021, and it may save data to BGT2022. Therefore it is often useful to automatically calculate years before and after the file group year, for use in file group variables and table variables.

Axiom provides four built-in variables to perform these calculations:

- FileGroupYearPlusX
- FileGroupYearMinusX
- ShortFileGroupYearPlusX

- ShortFileGroupYearMinusX

When using these variables, change the X to the number of years to calculate before or after the file group year. For example, if the file group year is 2022 and you want to return the previous year, you would use FileGroupYearMinus1.

You might create a set of year-related variables for use in plan file column headers, and then also reference those year variables in your table variables. For example, if the file group year is 2022 you might have some variables and table variables as follows:

| Variable | Definition | Resolves To |
|---|---|---|
| CurrentFY | {FileGroupYearMinus1} | 2021 |
| CurrentFYShort | {ShortFileGroupYearMinus1} | 21 |

| Table Variable | Definition | Resolves To |
|---|---|---|
| BGTData | BGT{FileGroupYear} | BGT2022 |
| GLDataCurrentFY | GL{CurrentFY} | GL2021 |
| GLDataLastFY | GL{LastFY} | GL2020 |

This approach is useful when changing the file group year as part of a rollover process. When the file group year is changed from 2022 to 2023, then the results of the year variables and table variables will update accordingly.

## Summary of built-in year variables

The following year-related variables are available for all file groups. These built-in variables can be used within other user-defined variables, and/or the values for these variables can be returned directly using the GetFileGroupVariable function.

| Variable | Description |
|---|---|
| `{FileGroupYear}` | The year for the file group as defined in the file group properties. This must be a four-digit year entry such as 2022.<br><br>All other year-related variables are dependent on a valid entry for the file group year. |
| `{FileGroupYearMinusX}` | Automatically calculates and returns years prior to the file group year, given a value for X. For example, enter 1 for X if you want to return the value of the previous year (file group year minus 1). If FileGroupYear is 2022, then FileGroupYearMinus1 is 2021. |

| Variable | Description |
| --- | --- |
| {FileGroupYearPlusX} | Automatically calculates and returns years after the file group year, given a value for X. For example, enter 1 for X if you want to return the value of the next year (file group year plus 1). If FileGroupYear is 2022, then FileGroupYearPlus1 is 2023. |
| {ShortFileGroupYear} | Automatically calculates and returns the short version of the file group year. For example, if FileGroupYear is 2022, then ShortFileGroupYear is 22. |
| {ShortFileGroupYearMinusX} | Automatically calculates and returns the short version of years prior to the file group year, given a value for X. For example, enter 1 for X if you want to return the value of the previous year (short file group year minus 1). If ShortFileGroupYear is 22, then ShortFileGroupYearMinus1 is 21. |
| {ShortFileGroupYearPlusX} | Automatically calculates and returns the short version of years after the file group year, given a value for X. For example, enter 1 for X if you want to return the value of the next year (short file group year plus 1). If ShortFileGroupYear is 22, then ShortFileGroupYearPlus1 is 23. |

4

# On-Demand File Groups

On-demand file groups allow end users to create new plan files on-the-fly. This feature is intended for planning processes where the list of plan codes is not predefined.

For example, planning processes such as budgeting and forecasting use a predefined list of plan codes, such as departments or divisions. In this scenario, an administrator creates plan files for all of the plan codes at the start of the process, and then rolls them out to end users such as department managers. Departments may be added or removed over time in response to organizational changes, but this process is managed by an administrator, not by end users.

Other planning processes such as capital project planning or strategic planning may not have a predefined list of plan codes. In this scenario, you may want users to be able to create plan files for new capital projects or new strategic initiatives as the need arises. If the file group is an on-demand file group, users can click a button to add a new file to the file group. Behind the scenes, Axiom generates a plan code and adds it to the plan code table, and then creates the plan file.

## Configuring a file group as an on-demand file group

If you want a file group to allow creating plan files "on demand," the file group properties must be configured as follows.

### ▶ Plan code table

The plan code table for the file group must be an *identity table*, meaning a table where the key column uses the **Identity** data type. When selecting the plan code table, identity tables are indicated in the drop-down list with a special icon  . Once an identity table is selected, the file group is automatically flagged as an on-demand file group.

Identity columns allow for the automatic generation of plan codes in a sequence. When a new plan file is created, a new record is created in the plan code table and the next number for the identity column is automatically generated. The sequence always starts at 1 and then increments using whole numbers. When using an identity column, you cannot manually define plan codes. For more information on identity columns, see the *System Administration Guide*.

**NOTE:** With standard file groups, the same reference table may be the plan code table for several file groups, such as the DEPT table. However, with on-demand file groups, each file group should have its own dedicated identity table. It is not recommended to re-use the same identity table for multiple file groups.

When using an identity table, it is common to also maintain an "alternate code" column in the plan code table. For example, you might generate a more meaningful code for each plan file, such as 43000_2021_16 (Dept_Year_Identity), and store this in a column called RequestID. If you want this alternate code to be the primary code that displays to end users (instead of the less meaningful identity value), then there are a number of settings that you can adjust in the file group properties to do this:

- On the **File Group** tab, you can change the **Tab Column** to be the alternate code column instead of the identity column. This means that the alternate code will display on file tabs for open plan files.
- On the **Display Columns** tab, you can edit the **Sorting columns** to remove the identity column and instead sort by the alternate code column. This means that lists of plan files will sort using the alternate code instead of the identity value.
- On the **Display Columns** tab, you can edit the **Display Columns** to add the alternate code column and place it at the top of the list, so that it will be the first value that users see when viewing lists of plan files. You can choose to remove the identity column so that it does not display to users at all, or you can move it to an appropriate location in the display columns so that it presents as "additional information" to users (like other grouping columns that may be displayed).

Keep in mind that all of these optional adjustments only affect the display of plan files. The identity value is still the official plan code for each plan file, if you need to reference that value in any system features such as the GetPlanFilePath function.

### On-demand options

Once the file group has been flagged as an on-demand file group through selection of the plan code table, a set of on-demand options becomes available in the file group properties. Most of these options are located on the **Options** tab, in the **On Demand Options** section. These options should be completed as follows:

| Option | Description |
| --- | --- |
| Disable Clone Existing Plan Files | Optional. Select this check box if you do not want to allow users to create new plan files by cloning existing plan files. By default this option is not selected, which means that the "clone existing" feature is enabled and available.<br><br>If this option is selected, then the **Clone selected item** command is hidden in the Open Plan Files dialog.<br><br>**NOTE:** This option is located in the **Plan File Options** section instead of the **On Demand Options** section. However, it only displays for on-demand file groups. |
| Plan File Process | Specify a plan file process for the file group. You can select any plan file process definition located in the Processes folder for the file group. Standard process definitions cannot be used. This option is required if you want to use a plan file process with the on-demand file group.<br><br>This designated process is used as follows:<br>• As new plan files are created in the file group, the plan files will be automatically started in the process designated here.<br>• Process Summary components for Axiom forms use the process designated here. You point the component to a file group or a file group alias, and it automatically uses the designated process for the file group.<br><br>For more information on using plan file processes with on-demand file groups, see Using plan file processes with on-demand file groups.<br><br>**NOTE:** This option is located in the **Process Options** section instead of the **On Demand Options** section, because standard file groups can also use this option for the Process Summary component. |
| Add File Message | Enter the text that you want to display to users to create a new plan file, such as "Add new file" or "Add new capital request".<br><br>This text will display in the top right-hand corner of the Open Plan Files dialog in the Excel Client and the Windows Client.<br><br>This text is required, even if end users will not be using this option to create plan files (for example, if all end users will use the Web Client). |

| Option | Description |
|---|---|
| Add File Form | Optional. Specify an Axiom form to use as the "input form" to collect starting values for the plan code table when creating a new plan file. You can select any form-enabled file stored in the Utilities folder for the file group (recommended), or in the Reports Library.<br><br>This form is used when a user clicks the "add file" option for the on-demand file group within the Axiom Excel Client or Windows Client. For more information on using this option, see Using an Axiom form as an "add file" dialog.<br><br>This option has no effect on the Web Client. If your users will be using the Web Client to create new plan files, then you must still create a form to facilitate plan file creation, but there is no built-in programmatic way to launch that particular form from the Web Client. It is assumed that users will either manually open the necessary form, or you will provide links to the necessary form, such as from a home page. |
| Clone File Form | Optional. Specify an Axiom form to use as the "input form" to collect starting values for the plan code table when cloning an existing plan file. You can select any form-enabled file stored in the Utilities folder for the file group (recommended), or in the Reports Library.<br><br>This form is used when a user clicks the **Clone selected item** option for the on-demand file group within the Axiom Excel Client or Windows Client. For more information on using this option, see Using an Axiom form as an "add file" dialog.<br><br>**NOTE:** If **Disable Clone Existing Plan Files** is selected (see below), then the Clone File Form setting is disabled. |

## Other file group settings

- You can use a **Default Template** and/or a **Template Column**. If you use a template column, then you must pass the name of the appropriate template to the plan code table when the new plan file is created. For more information, see Template options for on-demand file groups.
- In most cases, the **Allow Generation of Plan Files from Templates** option should remain unchecked. This option is unchecked by default when the file group is flagged as an on-demand file group. This option controls access to the Create Plan Files utility, which is normally not used for on-demand file groups.
- The file group can optionally use a **Show On List Column**. If defined, the Show On List column is automatically set to True when new plan files are created. For more information, see Using a Show On List column.

# Collecting values for the plan code table when creating an on-demand plan file

When a user creates a new plan file for an on-demand file group, a new record is automatically generated for that file in the plan code table. By default, only one column in this new record is populated with a non-default value: the identity key column, which contains the automatically generated ID number.

In many cases, you may have other columns in the table that need to be populated with non-default values when the new record is created, such as:

- Validated columns. For example, the plan code table could contain a column Dept that has a lookup assignment of Dept.Dept. When the new record is created, most likely you want the user to select a department (or you want to determine the appropriate department using a formula). If no value is specified when creating the new record, the default value for the column will be used. The specified or default value must be valid in the context of the assigned lookup column.
- Alternate key columns. Alternate key columns must be populated with a unique value for each record.
- Template assignment column. If the file group has an assigned Template column, then this column must be populated when the new record is created.
- Any other columns used to drive system settings or processes, such as columns for process ownership or security filters. In many cases these columns must be populated when the record is created, or else the associated feature will not work as expected and plan file creation may fail.

### Options to collect starting values

There are several ways to apply these starting values, depending on the environment used to create new plan files:

| Environment | Options |
|---|---|
| Desktop Client (Excel Client or Windows Client) | • By default, Axiom will automatically prompt the user to select values for any validated columns and alternate key columns in the plan code table. If a Template column has been specified for the file group, the user will also be prompted to select a template. This is built-in functionality that requires no additional setup. However, when using the built-in dialog, it is not possible to add other columns to the dialog or to omit certain validated columns.<br>• If you need to collect values for other columns, or if you just want greater control over the design of the dialog that is presented to end users, then you can create an Axiom form to serve as the "input form" for the plan file creation process. This form can be configured to collect all values that need to be passed along to the plan code table using the Add Plan File command. If this form is specified as the **Add File Form** in the file group properties, then when the user creates a new plan file this form will be displayed as a "dialog" to gather the user inputs, instead of the built-in prompt. |
| Web Client | This environment does not have any built-in functionality to prompt for column values. You must design an Axiom form to serve as the "input form" for the plan file creation process. This form can be configured to collect all values that need to be passed along to the plan code table using the Add Plan File command. |

For more details on the user experience for these environments, see How end users work with on-demand plan files.

For example, imagine that the on-demand file group is for creating capital planning requests, and you want all users to specify a department to associate with the request. You create a validated column named Dept in the plan code table, and you assign that column to a lookup column of Dept.Dept.

When Axiom creates a new record in the plan code table as part of the "add plan file" process, it must have a valid value to place in that Dept column.

- If you are using the built-in dialog in the Desktop Client, Axiom will automatically prompt the user to select a department. That department will then be used when creating the new record.
- If you are using an Axiom form (in any client), then the form designer can configure the form to pass a department value to the "add plan file" process. The form can prompt the user to select a department for this purpose, or the appropriate department can be derived using some logic in the source file for the form. The form designer can also choose to omit the Dept column from the "add plan file" process, in which case the column's default value will be used (as defined in the column properties).

Now imagine the same scenario, except this time the plan code table also has a non-validated column named Owner that you want to use for security filters. The column used for security filters must be populated when the record is created, or else the user will not be able to create or open the new plan file because the current blank value in the column does not meet the user's security filter (see Security considerations for on-demand file groups). In this case, for all user environments you must use an Axiom form to create the plan file. The built-in dialog cannot be used, because there is no way to customize this dialog to add the non-validated Owner column.

# How end users work with on-demand plan files

The behavior for creating and opening plan files for on-demand file groups depends on the client environment.

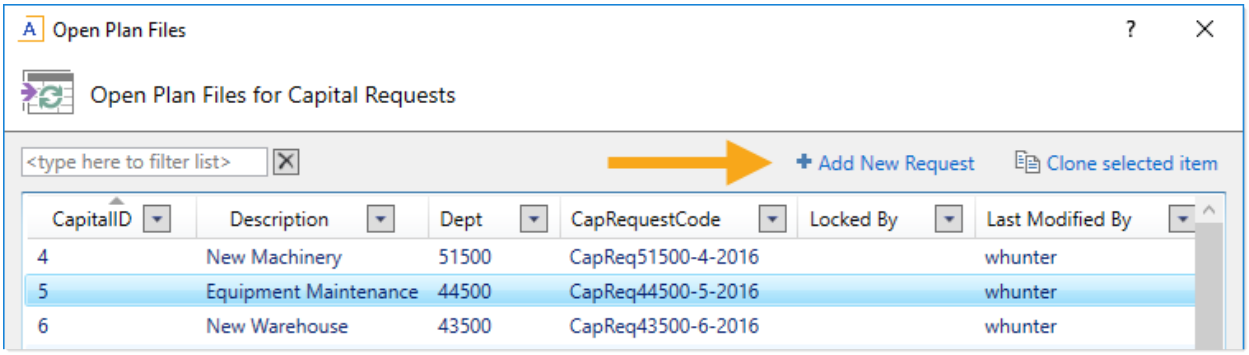
## Opening existing plan files

To open existing plan files in the Desktop Client (Excel Client or Windows Client), users use the **Open Plan Files** dialog as normal.

The Open Plan Files dialog does not apply to the Web Client. To open existing plan files in the Web Client, you must configure an Axiom form to present the appropriate hyperlinks to the user, or the user can open files from the "built-in" list of available forms in the Web Client.

## Creating new plan files—Windows Client and Excel Client

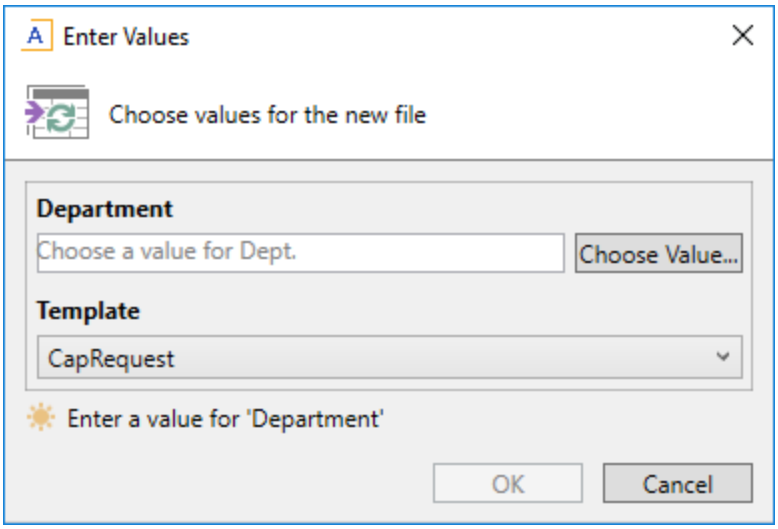
If a file group is configured to allow adding plan files on demand, the **Open Plan Files** dialog now contains a command to create a plan file. The name of this command is defined in the file group settings, so it will vary for each file group. In the example below, the command is named "Add New Request."




To create a plan file, the user clicks on this link. From here there are two different plan file creation paths, depending on whether an Axiom form has been specified as the **Add File Form** in the file group properties.

### Scenario 1: Default behavior

The user will be prompted to specify values for "required" columns in the plan code table. This includes alternate key columns, validated columns, and the file group Template column (if used).




*Example dialog to prompt the user for values*

**NOTE:** The **Hierarchy Display Name** for the column, if defined, is used in this dialog instead of the regular column name.

Once any required values have been specified, Axiom automatically creates a new record in the plan code table, using the next number in the sequence as well as the column values collected from the user (if applicable). A plan file is then automatically created for this new code, using either the default template assigned to the file group, or the template listed in the template column. The new plan file opens and the user can work with it as normal.

### Scenario 2: Using an Axiom form to create the plan file

If an Axiom form has been specified as the **Add File Form** in the file group properties, then this form is displayed as a "dialog" within the application. This form replaces the built-in dialog as described in the previous scenario, however, when using a form you have more leeway as to which values the user is prompted to select. The form must be configured with a Button component that uses the **Add Plan File** command.

When the user clicks on the button in the form, Axiom automatically creates a new record in the plan code table, using the next number in the sequence as well as the column values collected from the Axiom form. A plan file is then automatically created for this new code, using either the default template assigned to the file group, or the template listed in the template column.

The new plan file may or may not open at this point, depending on whether **Open plan file after creation** is enabled for the Add Plan File command. If opened, the plan file will open either as an Axiom form or as a spreadsheet, depending on whether the template used to create the plan file is forms-enabled or not.

### Creating plan files by cloning existing

As an alternative to creating a plan file from a template, users can also choose to clone existing plan files. To do this, the user selects the plan file in the Open Plan Files list, and then clicks **Clone selected item**. For example, the user may want to create a new capital or strategic project that is very similar to an existing project. In this case they can clone the existing project and then edit the copy, rather than starting from the default template. The starting values for the plan code table are collected as follows for the cloned plan file:

- By default, Axiom prompts the user to select values for any validated columns. The dialog uses the values from the cloned record as default values, but the user can choose to change the values for the new file. If the file group has an assigned Template column, the user will *not* be prompted to select a template—instead the template is recorded as the template used for the cloned plan file.
- Alternatively, you can specify a **Clone File Form** to collect the starting values. This form works like the Add File Form, but it should be designed slightly differently so that the form uses the values from the cloned record as default values. Additionally, the form should not prompt the user to select a template (if it does, the input will be ignored). Instead the template is recorded as the template used for the cloned plan file.

**NOTE:** If you do not want users to be able to clone plan files, this can be disabled on a per file group basis by selecting **Disable Clone Existing Plan Files** in the file group properties. See Configuring a file group as an on-demand file group.

Additionally, it is possible to customize the process for creating new plan files in the Excel Client or Windows Client. For example, you can place the **Add Plan File** command in a custom task pane or ribbon tab, in which case the command behaves as if the user had clicked the "add file" command in the Open Plan Files dialog. You can also use Axiom forms outside of the Add File Form context. For example, users can create new plan files from their form-enabled home page or from a stand-alone form dialog (in this case the behavior is the same as when using the Web Client).

## ▶ Creating new plan files—Web Client

When using the Web Client, you must create an Axiom form that users can use to create new plan files. The setup for this form is the same as the setup for the Add File Form, except in this environment the form will be presented as a web page instead of as a dialog. At minimum, the form must contain a Button component that the user clicks to create the new plan file. For more information, see the *Axiom Forms and Dashboards Guide*.

Users can access this designated web page as follows:

- You can add the Process Summary component to an Axiom form (typically the Home file), and enable the option **Show new item button**. Users can then click the plus icon in the component header to launch the designated Add File Form for the file group.
- You can add a link to the Plan File Directory page to an Axiom form (typically the Home file). Users can click the plus icon on the Plan File Directory page header to launch the designated **Add File Form** for the file group.
- You can place a direct link to the form used to create plan files on the users' form-enabled home page, or on a similar landing page. (In this case it is not necessary to designate the form as the Add File Form for the file group since you are linking to it directly, however, you may still wish to for other reasons.)

Typically, the plan files created via the Axiom form are also form-enabled and will subsequently open in the Web Client environment. However, it is also possible to have a "hybrid" configuration where an Axiom form is used to create plan files, but the resulting plan files are spreadsheets to be worked on later using the Windows Client or the Excel Client.

# Using an Axiom form as an "add file" dialog

When creating new plan files for an on-demand file group, you often want to collect "starting values" from end users. These values are saved to the plan code table when the new record is created. For more information on why you might need or want to collect these values, see Collecting values for the plan code table when creating an on-demand plan file.

If users are creating these new plan files in the Desktop Client, there are two options to collect these starting values. For more information on the end user experience for these options, see How end users work with on-demand plan files.

- By default, Axiom will prompt the user to define values for certain columns in the plan code table. This dialog is generated automatically and cannot be customized.
- Alternatively, you can design an Axiom form to collect these values, and designate this form as the "add file form" for the file group. Axiom will present this form to the user as a "dialog" in the application instead of using the default dialog.

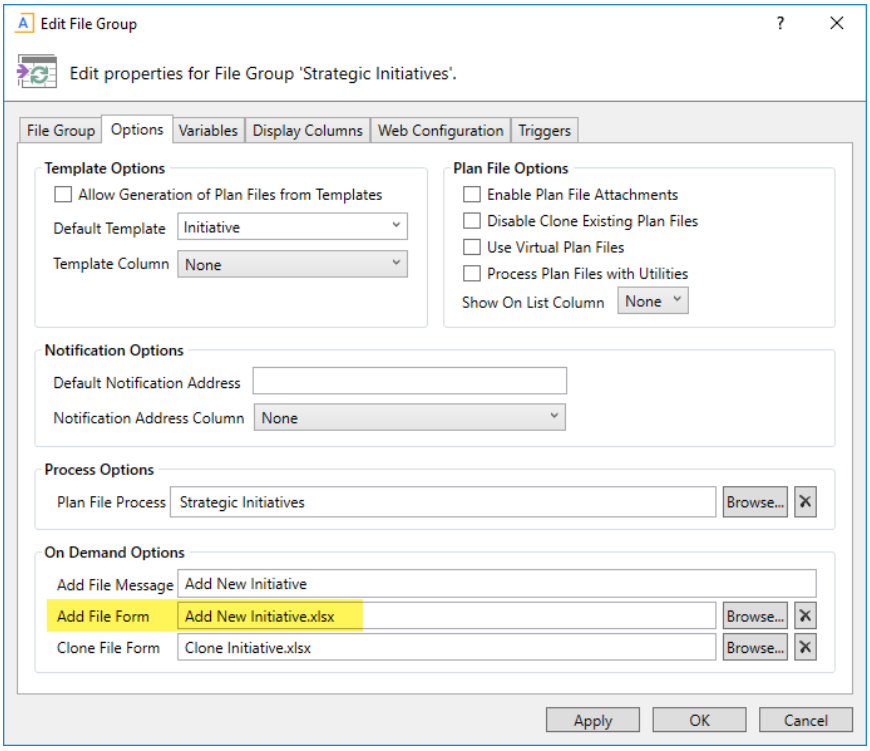
### Using an Add File Form to create new on-demand plan files

To use an Axiom form as dialog for an on-demand file group, you must do the following:

- Create the Axiom form to use as the dialog. In summary, the form must be set up as follows:
    - The form must collect the necessary values for columns in the plan code table. The designated Template column (if applicable to the file group) and any alternate key columns must be included. Other columns (including validated columns) can be included as needed, or omitted to use the default value defined for the column. The values can be obtained through user input or derived through other logic (such as using GetUserInfo to return the current user name).
    - The form must contain a Button component that uses the **Add Plan File** command. This command is used to pass the collected values to the plan code table and create the new record, and then create the new plan file. The command includes a configuration setting that determines whether the new plan file automatically opens after creation (in the majority of cases this should be enabled). It is also recommended to configure the button with a second command, **Close Dialog**, so that the dialog automatically closes after the new plan file is created.
    - If desired, you can define a form title (in the Form Properties) and this title will display as the dialog title. If no form title is defined, the title of the dialog will be **Add New Record**.

  For more information on how to set up this type of form, see the *Axiom Forms and Dashboards Guide*.
- Save the Axiom form as a utility file within the file group. It is also possible to save the form in the Reports Library, but it is recommended to save the file as a utility instead. Utility files reside within the file group and can stay in sync with it when the file group is cloned (this requires using the **Use Current File Group** setting in the Add Plan File command for the form).

- In the file group properties, on the **Options** tab, use the **Add File Form** setting to point to the Axiom form.




*Example file group configured with an Add File Form*

> **NOTE:** Users are not required to have any file permissions to the file designated as the Add File Form. If the user has the appropriate rights to create a new plan file for the file group, then the form will automatically display for the user in this context.

When a user clicks the "add new file" command in the Open Plan Files dialog, then the form designated as the Add File Form opens as a dialog. The user can complete the settings in the form, then click the button that is configured with the Add Plan File command. This will create the new record in the plan code table (including the user's inputs) and then create the new plan file. Whether the plan file opens or not depends on the configuration of the Add Plan File command. If a Close Dialog command is included on the button, the dialog closes automatically after the plan file is created (otherwise the user must manually close the dialog).

The Add File Form is also used automatically when users create on-demand plan files using other avenues, such as when using the Add Plan File command, or the option to add new plan files from the Process Summary component in Axiom forms, or the option to add new plan files from the Plan File Directory web page in the Web Client.

### Using a Clone File Form to clone existing on-demand plan files

In addition to creating new on-demand plan files from a template, users can also choose to clone an existing plan file. If you are using an Add File Form dialog to collect values when creating a new plan file, then you may also want to use a form dialog when a user clones an existing plan file. You can do this by designating a form as the **Clone File Form** in the file group properties (see screenshot in the previous section).

This form should be set up like the Add File Form. However, you may want to use the values from the cloned record as default values instead of having the user complete the inputs from scratch. You can do this by using the following reserved key in a GetFormState function within the Clone File Form:

```
=GetFormState("SourceID")
```

The SourceID key will return the ID of the plan file that is being cloned (this is the automatically-generated number from the Identity column). You can then use this information in GetData data lookups or GetData functions to return values for the cloned file. For example, imagine that all on-demand plan files are associated with a department, and you want to return the department that was associated with the plan file being cloned. You can do this as shown in the following example :

```
=GetData("RequestID.Dept","RequestID="&$D$10)
```

Where RequestID is the name of the plan code table and also the name of the identity key column, and the GetFormState("SourceID") function is in cell D10.

The SourceID value is passed to the Clone File Form when a user clicks **Clone Selected Item** in the Open Plan Files dialog.

Remember that if you are using a Template column with the on-demand file group, you should *not* prompt the user to specify a template when cloning. The recorded template must be the same template that was used for the original plan file being cloned. If you do prompt the user to select a template value and then attempt to pass that value to the plan code table, that value will be ignored.

# Template options for on-demand file groups

When using on-demand file groups, new plan files can be created based on the following templates:

- The **Default Template** specified for the file group
- The template listed in the **Template Column**, if a template column is specified
- An existing plan file can be cloned and used as the "template" for the new plan file (Desktop Client only)

If you want to use a template column for the file group, then the template column must be populated when the new record is created in the plan code table. If the template column is not populated when the new record is created, then when Axiom creates the new plan file it will find a blank value (empty string) in the template column, which means that the default template for the file group will always be used. (Although you could optionally define a template name as the default value for the column instead of using empty string, that would effectively be the same as using the default template for the file group—

meaning that all of the records would be assigned the same template.) For more information how column values can be populated when the plan file is created, see Collecting values for the plan code table when creating an on-demand plan file.

How the template column gets populated depends on the user environment for creating new plan files.

### Desktop Client (Excel Client or Windows Client)

If you are using the default functionality to create on-demand plan files in the Desktop Client, then Axiom will automatically prompt the user to select a template when creating a new plan file. The user will be prompted as follows:

- The template drop-down list will display on the same built-in dialog that is used to prompt the user to select values for validated columns.
- The drop-down list will contain all templates that are defined for the file group. There is no way to exclude a template from this list. Therefore if you are using this option, you should not keep any old or archived templates in your template folder. If only one template exists in the template folder, then the user will not be prompted and that template will be used.
- If a default template is also specified for the file group, that template will be selected by default.

The user's selection will be populated in the template column when the new record is created, and then that template will be used to create the plan file.

If you are using the "alternate" functionality where an Axiom form is used to collect user inputs instead of the default behavior (using the **Add File Form** setting for the file group), then that form must be configured to collect a template value and pass it to the template column. You can present the user with a list of templates to select from, or you can use logic within the source file for the form to determine the appropriate template.

### Web Client

The Web Client does not provide any built-in functionality to prompt the user to select a template. When you configure the Add Plan File command for the Axiom form that is being used to create new plan files, you must include the template column and set up the form to collect a value for that column.

If the form is not configured to pass a value to the template column, or if the value is blank, then the default template for the file group will be used.

For more information on configuring an Axiom form to create new plan files, see the *Axiom Forms and Dashboards Guide*.

# Using plan file processes with on-demand file groups

You can use process management to define a *plan file process* for on-demand plan files—meaning, a set of defined steps for plan files to progress through. This section details some design considerations to keep in mind when defining the plan file process definition for an on-demand file group.

As each new plan file is created in the on-demand file group, it will be automatically added to the process and will progress through each step. In order for newly created plan files to be automatically started in the process, the process must be set up as follows:

- The process must be created as a plan file process definition within the on-demand file group.
- The process must be designated as the **Plan File Process** in the file group properties, on the **Options** tab.
- The process must be active.

When configuring the process settings, note the following:

- Typically the ownership assignment for the first step is set to **Process Initiator**. In most cases this means that the owner is the user who created the file and therefore caused the plan file to be started in the process. Other ownership assignment options can be used, but if the creator is not the owner of the first step then the creator may be unable to create the file at all, or if the file is created they may be unable to save it (unless their security permissions already allow that level of access).
- In most cases, relative due dates should be used for each step. For example, in the first step, you could specify that each new plan file must be moved to the next step within 5 days. Then, in the next step, managers could have 7 days to approve or reject the plan file.

  However, you could use defined due dates if the cycle of creating and approving plan files is limited to a specific time period rather than an ongoing process. For example, users could have a month-long window to propose new projects, and then managers would have to review them and determine which are approved, by a specific due date.

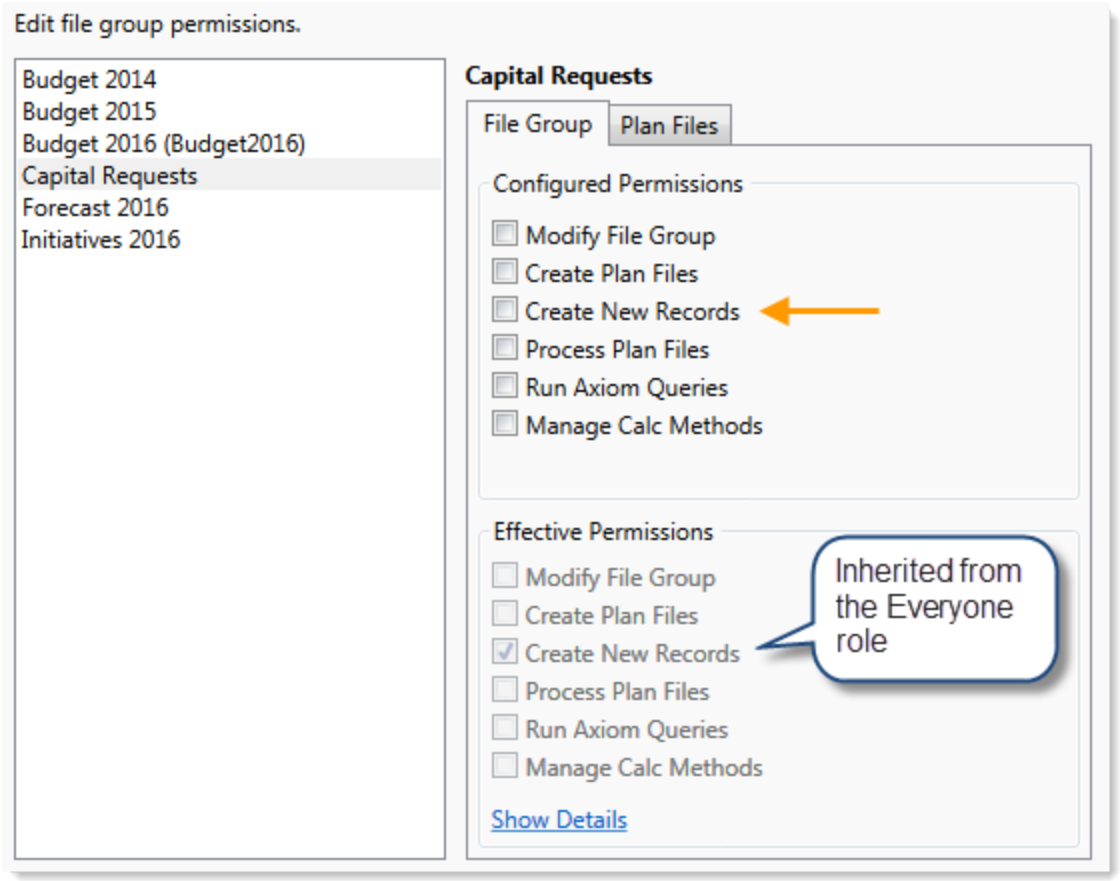
Plan file processes for on-demand file groups also have access to a special option that allows owners of approval steps to "deny" a request, meaning that the plan file is aborted in the process and does not progress further. This is intended for processes such as capital requests or purchase orders, where some requests need to be denied rather than temporarily rejected. For more information, see the *Plan File Process Guide*.

# Security considerations for on-demand file groups

The following additional security considerations apply when using on-demand file groups.

### Ability to create new plan files

For on-demand file groups, a user's ability to create new plan files depends on the **Create New Records** security permission. This permission is located on the **File Group** sub-tab of the File Groups tab in security, and is only available for on-demand file groups. However, in most cases you do not need to configure this permission in security, because it is already granted via the Everyone role.




By default, when you create a new on-demand file group, the Create New Records permission is enabled on the Everyone role for that file group. Effectively, this means that if a user is otherwise granted access to plan files in that file group (on the **Plan Files** sub-tab), then that user will be able to create new plan files as well. The Everyone role is used because typically if a user has access to the on-demand file group, then you also want the user to be able to create new plan files in the file group. However, if instead you want some users to be able to create new plan files and other users to only access existing files, then you can remove the permission from the Everyone role and instead assign it to individual users and roles as appropriate.

**NOTE:** The terminology of the permission refers to creating new *records* instead of creating new plan files in order to capture the unique "on-demand" process of creating a new record and a new plan file in the same action. By contrast, the **Create Plan Files** permission refers to the ability to create plan files for *existing* records using the Create Plan Files utility. The Create Plan Files permission is not necessary to allow end users to create new records and plan files for on-demand file groups.

Keep in mind that once a user creates a new plan file, in most cases you also want the user to be able to save it. One way to accomplish this is to grant the user the appropriate level of plan file access directly (**Read/Write** and **Allow Save Data**). Alternatively, if you are using a plan file process, then you can set the user's access to **No Access** or **Read-Only** and allow the process to "elevate" the user's access as appropriate to the plan file. For this to work, the first step of the process must be configured so that the assigned owner is the process initiator / plan file creator.

**NOTE:** If you want a user to be able to create a new on-demand plan file by cloning another plan file, then that user must have at least **Read-Only** access to the plan files in Security. Users with **No Access** plus **Interacts with Process Management** can create new plan files, but cannot clone existing plan files.

### Using a plan file filter

When configuring a user's plan file permissions to an on-demand file group, you can opt to give the user access to all plan files in the file group, or define a filter. If you want to use a filter, then special considerations apply.

When defining a security filter for an on-demand file group, keep in mind that the column used to define the filter must be populated at the point when the new record is created in the plan code table. Otherwise, the user will not be able to create any plan files because their security filter will be applied at the point of creation, and at the point of creation the column will be blank. If Axiom detects that the user's security filter will not allow access to the new plan file, then the plan file creation is stopped (and the new record is not created in the table). For more information on this issue and how to collect starting values for the plan code table, see Collecting values for the plan code table when creating an on-demand plan file.

This limitation means that your security requirements may impact how you decide to configure the plan code table, and what approach you decide to adopt for end users to create new plan files. For example, imagine that your users use the Excel Client, and you want to set security filters based on department. You can create a column named Dept in the plan code table, and configure that column to lookup to Dept.Dept. Since the default behavior in the Excel Client is to automatically prompt users to define values for validated columns, then you know this column will be populated when the new record is created. So you can define security filters such as:

`PlanID.Dept=4200` to limit the user to plan files associated with Dept 4200

`Dept.Region='West'` to limit the user to plan files associated with the West region

However, now imagine that you want to base the security filter on a column named Owner, which is *not* a validated column. Now the default method of plan file creation will not work, because it will not populate the Owner column. In this case, you must instead use an Axiom form to collect the starting values for the plan code table. You might set up the form with a GetUserInfo function that returns the current user name, and then configure the Add Plan File command to pass that value to the Owner column. Now, you can have a security filter such as `PlanID.Owner='jdoe'`, because the Owner column will now be populated when the record is created.

If your system design is such that you cannot populate a particular column when the record is created, and you must use that column to define security filters, then you can work around this by adding the "null" case to the filter. For example, instead of using `PlanID.Owner='jdoe'` as the filter, you would use `PlanID.Owner IN ('Jdoe','')` (where the empty single quotation marks indicate null or blank). This tells Axiom that the user can access any plan file that has either "jdoe" or a blank value in the Owner column.

# Deleting plan files from an on-demand file group

Because plan files for on-demand file groups are created "on-the-fly" instead of being generated from a predefined list of dimension elements, there is often a greater need to delete plan files for on-demand file groups. For example:

- After performing tests on the file group, the administrator wants to delete the plan files that were created during testing. This need is typically unique to on-demand file groups, because when working with standard file groups the administrator would simply use **Create Plan Files** to re-create all plan files instead of deleting them.
- End users may create plan files by mistake. Instead of hiding those plan files using a ShowOnList column, a cleaner approach is to delete them. Again, this need is unique to on-demand file groups because end users cannot create plan files for standard file groups.

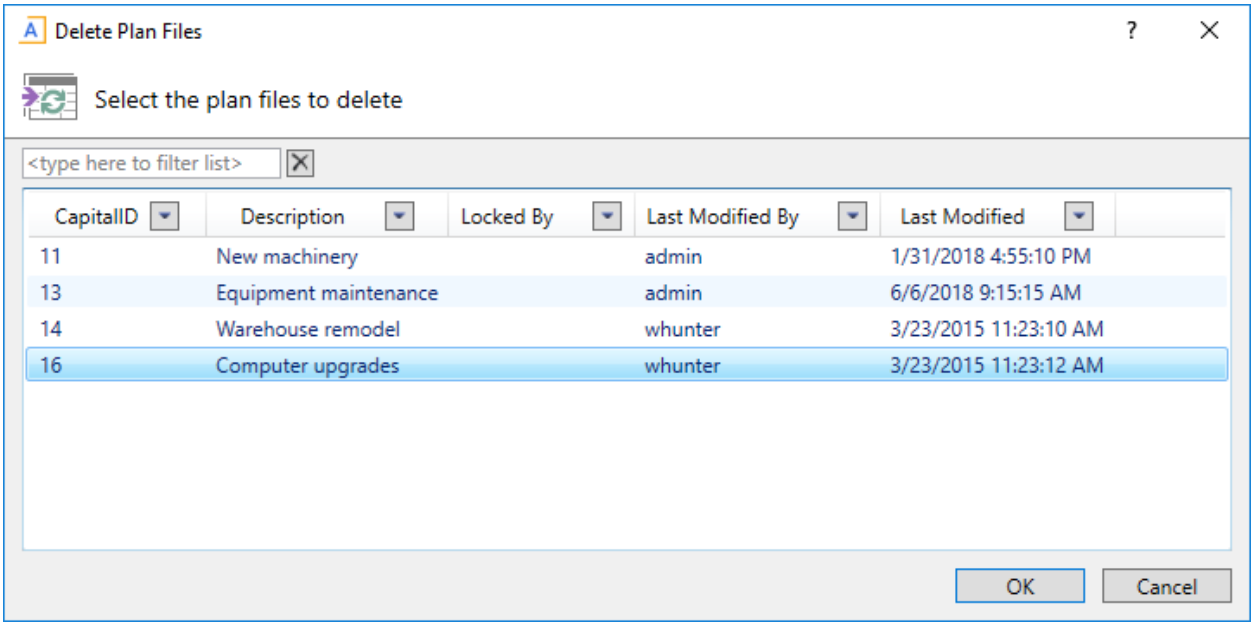
Although administrators can always delete plan files in Axiom Explorer, the simple act of deleting the plan file is not enough to clean up the system in this case. At minimum, the corresponding record in the plan code table must also be deleted. Data for the plan file may have been saved to other tables, and this data should be deleted as well. Additionally, there may be other artifacts in the system, such as plan file attachments and associated process tasks.

To streamline this process, you can use the **Delete Plan Files** command to delete plan files instead of performing these tasks manually. When a plan file is deleted using this command, all associated data and system artifacts are also deleted. You can also optionally expose this command to end users so that they can delete plan files that they created by accident.

Delete Plan Files is not exposed on any Axiom menus or dialogs by default. If you want to use the feature, you must add the command to a custom task pane or ribbon tab and then make that task pane or

ribbon tab available to the target users of the command. When you configure the command, you must specify a target on-demand file group for which to allow plan file deletion. For more information on creating custom task panes and ribbon tabs, see the *System Administration Guide*.

When a user executes the Delete Plan Files command from a custom task pane or ribbon tab, it opens the **Delete Plan Files** dialog. This dialog displays a list of the plan files that the user can delete. Generally speaking, administrators can delete any plan file, whereas end users can only delete plan files that they created (and where they have read/write access). The user can select one or more plan files to delete.




**NOTE:** There is no equivalent feature to delete plan files for a standard file group, because this is rarely necessary (**Create Plan Files** can be used to overwrite the existing files as long as the plan codes are still valid). If it is necessary to delete a plan file in a standard file group, then an administrator must manually delete the plan file in Axiom Explorer, and then manually delete any obsolete data in associated tables (the plan code table and any data tables that the file saved to).

## Who can use Delete Plan Files

- Administrators and users with the **Administer File Groups** permission can use the command to delete any plan file in the target file group.

- Non-administrators are treated as follows:
    - The command is visible to the user if the user has the **Create New Records** permission for the target file group, and the user has at least read-only access to some files in the file group.
    - The user is limited to deleting plan files where the user has read/write access AND the user is the plan file creator. If the user does not currently have any plan files that meet this criteria, an error message displays when the user attempts to use the Delete Plan Files command.
    - Permission "elevation" due to being the current task owner in a plan file process is honored for purposes of determining the user's current access level to plan files.

## What happens when a plan file is deleted

When a plan file is deleted using Delete Plan Files, the following occurs:

- The plan file is deleted.
- The corresponding identity record in the plan code table is deleted.
- Any corresponding data in linked tables is deleted. Linked tables are eligible for data deletion as follows:
    - The table must have a column with a lookup to the identity column in the plan code table.
    - This column must have **Is Cascade Delete** set to **True** to allow the automatic data deletion. This is set in the column properties.
    - All records linked to the deleted identity record in the plan code table will be deleted.
- Any corresponding plan file attachments are deleted (including the attachment folder).
- Any corresponding process tasks for the plan file are deleted.

Keep in mind that by default, **Is Cascade Delete** is set to **False**. You should set this to **True** if you want to use the Delete Plan Files command. Otherwise, plan file deletion will be prevented if the plan file has corresponding data in other tables.

**Example**

Imagine that a user creates a new capital request (ID 47) and saves it, which also causes data to be saved to the CPData2022 table. The CPData2022 table has a column CPREQ, with a lookup to the identity column in the plan code table. When the data is saved to CPData2022, it is saved with CPREQ as 47.

The user then realizes that they don't need this particular request, and they want to delete it using Delete Plan Files. If Is Cascade Delete is True for the CPREQ column, then deleting the plan file will also delete records associated with ID 47 in the CPData2022 table. If Is Cascade Delete is False, then the plan file and its plan code cannot be deleted using the Delete Plan Files command.

## Setup for Delete Plan Files

The following summarizes the setup steps to use Delete Plan Files for a file group:

1. Identify the tables that plan files in the file group save to, and determine whether the plan code is recorded in those tables. If it is, then enable **Is Cascade Delete** for the column in the table that looks up to the plan code identity column.

   For example, imagine that the plan code table is CPREQ2022 and the identity column is CPREQ. Plan files save data to CPData2022, and that table contains a column `CPData2022.CPREQ` that looks up to `CPREQ2022.CPREQ`. In order to delete associated records in CPData2022 when a plan file is deleted, Is Cascade Delete must be enabled for `CPData2022.CPREQ`.

2. Add the **Delete Plan Files** command adapter to a task pane and set it to the desired file group. You may need to create a new task pane for the command, or you may have an existing task pane that would be appropriate to add it to.

Users can then use the command in the task pane to delete plan files.

5

# Templates

Templates define the default file structure for plan files. Using Axiom queries and save-to-database tags, templates also define the data to be populated into plan files, and the data to be saved to the database.

Generally, each plan code has a plan file built from a template. You can have one or several templates for a file group, depending on your planning needs.

## About templates

Templates define the default structure for plan files. In addition to defining cosmetic features such as headings and freeze panes settings, templates use the following Axiom file features:

- Axiom queries, to dynamically populate sheets with data specific to each individual plan code
- Axiom functions, to look up driver data or other information such as account and department descriptions, or file group settings such as the current period
- Excel functions, for calculations and subtotals
- Save2DB tags, to define how data is saved to the database from the plan file
- Calc methods, to define calculation methods and formatting for rows of data
- Calc method controls, to control where calc methods can be used and which calc methods can be used
- Other specialized tags, such as GoTo tags and View tags

With the exception of calc methods, all of these features are standard to all Axiom files, and are discussed in the *Axiom File Setup Guide*.

Calc method libraries can only be used in templates and plan files. For details on managing calc method libraries and configuring calc method insertion points in templates, see Calc Method Libraries.

### Template sheet names

Template sheet names are linked to associated calc method libraries. Each sheet with the same name in a file group shares the same calc method library, and therefore must have the same column structure. Be very careful when changing the column structure of a sheet, as it will impact all calc methods in the associated library. See About calc method libraries.

Choose your template sheet names carefully. If a sheet name is changed, that sheet will lose access to its associated calc method library, and there is no way to "move" the existing calc method library to the new sheet name. If you must change a sheet name, you will need to bring all calc methods into the sheet first, rename the sheet, and then save each calc method back to the new library.

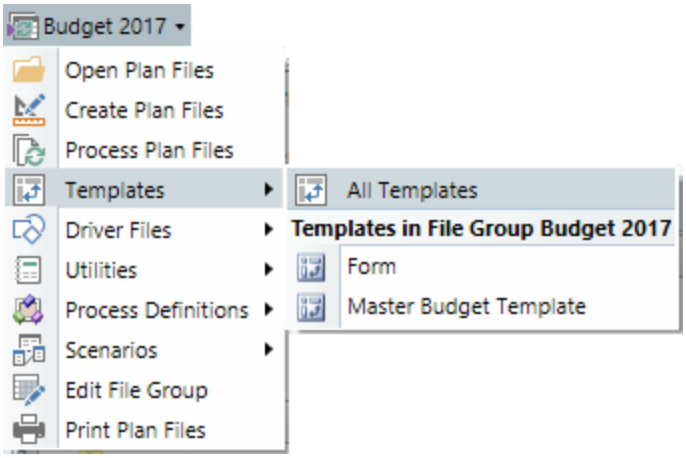
### Creating plan files from templates

To create plan files from templates, use the **Create Plan Files** utility. This feature can be used to selectively create plan files for individual plan codes (during preparation and testing phases), or to create plan files for all plan codes (when you are ready to roll out the plan files to end users). See Creating plan files.

If you are using an on-demand file group, then the **Create Plan Files** utility does not apply. Instead, users create new plan files as needed, using either the default template (as defined for the file group) or by cloning an existing plan file. For more information, see On-Demand File Groups.

# Managing templates

You can manage templates for a file group by using the file group buttons on the **Axiom** tab or by using the Explorer task pane. If you have rights to access templates, then clicking the file group button brings up the file group management menu.




By default, only administrators can create, edit, and delete templates. Non-admin users can be granted permission to access template files by using the **Files** tab in Security.

If the file group was created by copying an existing file group, copies of the templates may have been included as part of the copy operation. In this case the templates must be edited as needed for the new file group.

## Creating a template

All new templates are based on the template default file defined for the system. In Axiom Explorer, the template default file is located in `\Axiom\Axiom System\Document Templates\File Group Templates`. This file can be customized for the system if desired. Only one default file can be present in this folder.

By default, only administrators can create new templates. If you want a non-admin user to be able to create templates for a file group, then that user must have **Read/Write** access to the file group's **Templates** folder, as defined on the **Files** tab of the **Security Management** dialog.

**NOTE:** This topic explains how to create a new template file. For information on how to set up Axiom file features in the template, see the *Axiom File Setup Guide*.

**To create a new blank template:**

1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Templates > All Templates**.

   The **Template Files** dialog opens, listing the templates for the file group.

   **TIP:** You can also access this dialog from the Explorer task pane, by right-clicking the **Templates** node and selecting **All Templates**.

2. At the bottom of the dialog, click **Create**.

   The **Create File Group Template** dialog opens.

3. In the **Name** box, type a name for the new template, and then click **OK**.

   The template name can be anything you like. If you have multiple templates in a file group, the name should be descriptive of the template's intended use, so that you can easily distinguish the templates when editing them or assigning them to plan files.

   For example, if the file group is for budgeting and you have different templates for overhead departments versus revenue departments, you might name one template "Overhead" and one template "Revenue."

The new template is now available to be configured for the file group. The default template contains several sample sheets with sample formatting. You can adapt these sheets to your use, or you can delete the formatting and/or the sheets as desired.

### Creating a template by copying an existing template

In many cases, it may be easier to create a new template by copying an existing template. If you are an administrator, you can use the Explorer task pane or Axiom Explorer to copy or import files into the Templates folder of the file group: `\Axiom\File Groups\<FileGroupName>\Templates`.

You can also open the template file that you want to copy, and then use **Save As (Repository)** to save it to a template folder where you have read/write permissions.

### Editing a template

You can edit a template until you have created plan files based on that template. Once plan files have been created, any further changes to the template will not be reflected in those plan files, unless you re-create the plan files (or unless you are using virtual plan files).

**IMPORTANT:** You should be careful not to make any significant changes to a template once plan files are created, because the template is referenced by Process Plan Files and Print Plan Files to obtain lists of Axiom queries and print views respectively. Therefore you should not rename, delete, or otherwise edit any Axiom queries or print views in a template unless you plan to re-create plan files, or unless the existing plan files are no longer active and will not be used by these utilities.

**To open a template for editing:**

- On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu. Click **Templates**, then click the name of the specific template or click **All Templates** to select a template from the **Template Files** dialog.

  **TIP:** You can also open a template from the Explorer task pane or Axiom Explorer.

The template opens for editing. For more information, see Working with templates.

### Deleting a template

A template cannot be deleted if existing plan files were built using it, because the source template must be referenced when performing actions such as **Process Plan Files**.

**To delete a template:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The Axiom Explorer dialog opens, with the focus on the **File Groups** section of the treeview.

   **TIP:** You can also use the Explorer task pane to delete a template.

2. Expand the desired file group, and then select or expand the **Templates** node.
3. Right-click the template and then select **Delete**.

# Working with templates

Templates are the source documents for plan files. When working with templates, you set up features that are intended for eventual use in plan files, not within the template itself.

For example, most templates use Axiom queries to bring in data for each individual plan file. This Axiom query is only intended to be used within the resulting plan files, not within the template. The template itself should be kept clean of any department-specific data, so that it does not get accidentally left in the file and then copied into every individual plan file. To test template setup, you should create plan files based on the template and perform all feature testing in those plan files.

Note the following feature behavior differences when working in templates:

- Axiom queries that are set to refresh on open do not refresh when the template is opened. Additionally, formulas do not refresh automatically when the template is opened.
- Sheet protection, sheet visibility, and freeze panes settings as defined on the Control Sheet are not applied when the template is opened. This is intended to make it easier for you to work in these sheets without the added step of removing protection, unhiding, etc. All of these settings will be applied as expected in the resulting plan files (including workbook protection).
- The Control Sheet is visible while working in a template. When plan files are built from the template, the Control Sheet is always hidden by default in the resulting plan files. You do not have to hide the Control Sheet in order to hide it in plan files.
- Action codes are not processed automatically within templates. If you want to test the action code setup for a template, it is recommended to build a plan file and test it there. However, if you must test the setup within the template, you can use the **Process** command on the Axiom Designer ribbon tab to process action codes within a template.

## Editing template workbook structure

Templates can have as many sheets as needed to meet the planning goals of the file group. Sheets can be used for various purposes, such as:

- Instructions sheets to communicate instructions and contact information to end users
- Summary sheets displaying summarized data, for information purposes
- Planning sheets where end users input planning data
- Work sheets to support features such as drop-down lists on other sheets, or to hold notes and side calculations for end users

You can add and remove template sheets as needed using standard spreadsheet functionality. Keep in mind the following:

- If a new sheet will use a calc method library, give that sheet a unique name unless it is intended to share a calc method library with sheets in other templates in the file group. For example, if one template has a sheet named Capital, do not use that name for a sheet in another template unless those sheets will have the exact same structure and should share the calc method library.

  If a sheet does not use calc methods, then it can share the same name across templates. For example, all templates can have a sheet named Instructions, and that sheet can be structured differently in various templates, as long as the Instructions sheet does not use calc methods.

- Don't change the name of a sheet if calc methods exist for that sheet. Calc method libraries are stored based on sheet name. If you change the name of a sheet that has calc methods, that sheet will no longer have access to the calc methods.

  If this occurs, you can restore access to the calc methods by renaming the sheet back to its original name. If you must rename the sheet, you will need to bring all calc methods into the sheet first, rename the sheet, then save all calc methods back to the new library.

Workbook protection is always applied to plan files. When plan files are created based on the template, end users will be unable to delete sheets or add new sheets.

## Design considerations for template sheets

**NOTE:** This section assumes that you are using spreadsheet plan files. If you are designing form-enabled plan files, please see the *Axiom Forms and Dashboards Guide* for more information on designing forms in general, as well as planning-specific information.

Template sheets are free-format. You can structure the sheets however you like, but most templates are structured as follows:

- Work rows at the top of the template to hold Axiom row controls (such as save-to-database and Axiom queries), as well as other calculations and information that end users do not need to interact with. This area is typically "hidden" from end users by using the freeze panes setting on the Control Sheet.
- Work columns to the immediate left and to the far right, to hold Axiom column controls (such as save-to-database, Axiom queries, and calc method insertion controls). The work columns to the left are typically "hidden" from end users by using the freeze panes setting on the control sheet. The work columns to the right are placed past any columns that end users work with.
- The main planning area where users review the plan and complete planning inputs. This section usually contains a header area that is locked onscreen via freeze panes, with information such as the relevant plan code and column titles. The planning rows may be hard-coded (rows that are defined in the template and copied into all plan files built from the template), or variable (rows that are brought in via Axiom query, specific to the current plan code).

### Protecting sheets

Most template sheets are protected so that when plan files are created users can only make changes in designated cells. This sheet protection must be configured on the Control Sheet—do not manually apply protection to the sheet. Although the sheet protection is not applied while working in the template, the protection will be applied in the resulting plan files.

### Hiding sheets

When plan files are created based on the template, all sheets in the template will be visible to end users except those that are explicitly hidden (and the Control Sheet, which is always hidden in plan files). If the template has work sheets that users do not need to interact with, those sheets should be configured as hidden on the Control Sheet. Although these configured sheets will not be hidden while working in the template, they will be hidden in the resulting plan files.

### Reserved rows and columns

Rows 1 and 2 of a template sheet are reserved for template validation. You should not manually place anything in these rows or modify any existing contents. Template design should start at row 3. For more information, see Template validation.

**NOTE:** Template validation controls are optional, but we strongly recommend that you use them, to help ensure the integrity of your data, and to provide enhanced features for template and calc method management. However, if you have templates where you have placed content in row 1, then Axiom will not attempt to enforce the validation controls, and the template will still work.

## Changing the column structure of a template sheet

If a template sheet uses calc methods, then the column structure of the sheet is interdependent with:

- The associated calc method library for the sheet name
- Other template sheets that use the same sheet name (within the file group)

Because of these dependencies, you should be very careful when editing the column structure of a sheet in a template. If calc methods have already been created for a particular sheet, then inserting or deleting a column may cause a mismatch between the sheet structure and the associated calc method library.

Additionally, if other templates in the file group share the same sheet name, the sheets in those templates must have the same sheet structure, so that the calc method library is in synch with all related sheets.

Ideally, you should finalize sheet structure as much as possible before saving calc methods to the library, or before copying the template to make additional templates. If you need to make a change to the sheet structure once calc methods have been saved, you should:

1. Bring all existing calc methods into the template sheet *before* making any structure changes. Calc method administrators can use **File Options > Add Row(s) > Insert Calc Method(s)** to select all calc methods and insert them at once.
2. Make the sheet structure changes, including related changes to the calc methods as appropriate.
3. Use **Advanced > CM Library > Replace Multiple Calc Methods** to update all calc methods for the changes.

**NOTE:** At this point, if you are using template validation, you will be warned that a column layout change has been made to the template. Don't worry about the warning—you are in the process of resolving the issue and you will dismiss it after making all changes. Continue with the calc method replacement.

4. If other templates in the file group use the same sheet name, then make the same sheet structure changes to those templates. (You do not need to do anything with the calc methods in this case, because the calc methods have already been updated for the structure changes.)
5. Once all updates have been made to the calc method library and to all impacted templates, reset the column layout data: **Advanced > CM Library > Reset Column Layout Data**. This only applies if you are using template validation.

## Editing templates after plan files have been created

Generally speaking, you should not edit a template once plan files have been created with that template, unless you plan to rebuild the plan files. Once plan files have been created, any further changes to the template are not copied to existing plan files.

Some plan file utilities look to the source template in order to provide a list of choices for performing actions on plan files. In this case, the choices in the template must match the choices in the plan files. Because of this:

- You should not delete, rename, or otherwise edit Axiom queries in the template once plan files have been created, because the **Process Plan Files** utility uses the source template to present the list of Axiom queries to process.
- You should not delete, rename, or otherwise edit print views in the template once plan files have been created, because the Print Plan Files utility uses the source template to present the list of views to print.

However, if the file group is an on-demand file group, then you might want to edit a template even though plan files exist, so that the change can apply to new plan files created for the file group going forward. In this case it might not matter that the older plan files do not have the change. Alternatively you could create a new template to incorporate the change, and then edit the template assignments for the on-demand file group so that the new template is used going forward.

**NOTE:** This restriction does not apply when using virtual plan files with a file group, because each time a plan file is accessed, it is automatically rebuilt using the latest copy of the template. However, in this case you must be careful when making changes to templates, because if a user accesses a plan file while you are in the middle of making the change, the plan file will be built using this "in progress" copy. Instead you should make your change in a test copy of the template, verify that the change works as expected, then import the new copy of the template over the old copy (to preserve the template name and ID).

# Saving a template

When you save a template, the template file is updated in the Axiom file system.

**To save a template:**

- On the **Axiom** tab, in the **File Options** group, click **Save**.

**NOTE:** Saving data to the database is ignored for plan templates. Even though templates have Control Sheets with save-to-database processes enabled, those processes are intended to be used in the resulting plan files, not the template. In order to test the save-to-database settings in a template, you must build a plan file from the template and then perform the save. Although **Save > Save File Only** is available in templates, it has the same effect as the default **Save**.

## Save validations

The following validation checks are made when you save a template:

- The save process checks to make sure no sheets in the file have been manually protected. If a protected sheet is found, Axiom informs you that the protection will be removed before saving and asks you if you want to continue. To protect sheets, you must configure the sheet protection on the Control Sheet instead of manually applying protection.
- The save process checks for any cross-file links in the file. If any are found, a warning message is displayed with the details of where the links are located. The save is not prevented, but we strongly recommend removing these links before saving. Cross-file links are not supported, and are very likely to break and cause issues in the resulting plan files.

  In most cases these links are inadvertently created from copying and pasting content from another file. In this case the links should be updated to locations within the current file or removed as appropriate.
- The save process checks the used range of the file and warns you if it goes out to column XFD. Using the entire column range of the spreadsheet can cause issues with calc methods and other features. In virtually all cases, this situation is accidental rather than intentional, and simply needs to be corrected.

## Saving a copy of a template

You may want to save a copy of a template to use as a starting point for a new template, or to keep it as an "archive copy" (in case you want to return to an earlier version of the file later). To do this, you can use **Save As (Repository)** to save a copy of the file within the template folder for the file group, or to a different location. You can also use Axiom Explorer to create a copy of the file.

**NOTE:** Use caution before saving "archive copies" of templates within the template folder for the file group. Keeping unused files in this folder can lead to confusion when setting up template assignments for plan files, and when resolving template validation issues.

If necessary, you can use **Save As (Local File)** to save a copy of the file "offline". You might do this so that you can import the file into another Axiom system, or so that you can send the file to Axiom Support for troubleshooting.

# Template validation

Axiom provides various features to validate file group templates and ease the process of template administration. For example, using template validation you can:

- Be alerted if there is a potential mismatch between template structure and the calc method library.
- Gain access to enhanced calc method management features, including the ability to update all calc methods in a library in batch, and identify specific calc methods used in a sheet.

If template validation is enabled for a template, then all related validation features are automatically available for that template. This section explains how to enable template validation, and explains how the related validation features work.

**NOTE:** Template validation controls are optional, but we strongly recommend that you use them, to help ensure the integrity of your data, and to provide enhanced features for template and calc method management.

## Enabling template validation

In order to use template validation, you must design your template sheets so that rows 1-2 are reserved for system-generated codes. These codes are used by Axiom to perform functions such as tracking sheet structure.

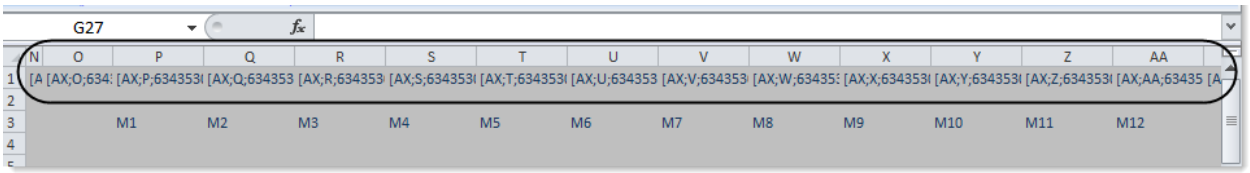
If row 1 is blank the first time that you create or insert a calc method in a template sheet, Axiom will place system-generated codes in row 1. The codes span the used area of the sheet, or through column AD, whichever is greater. Template validation is now enabled for that sheet of the template.

If you are using template validation, then you should not manually place any content in the reserved rows or modify any existing contents. If Row 1 contains content other than system-generated codes, then template validation is not applied and no validation features will be available.

**NOTES:**

- Currently, only row 1 is being actively used by Axiom for template validation purposes. Row 2 is reserved for future validation features. We recommend reserving both rows in your templates so that you can immediately take advantage of any future validation features, without needing to restructure your templates.
- Template validation controls are optional, but we strongly recommend that you use them, to help ensure the integrity of your data, and to provide enhanced features for template and calc method management.

Validation will apply to any sheet in the template that has an associated calc method library and uses the reserved rows. If a particular sheet in a template does not need to be validated, then you can place content in row one of that sheet, and then validation will not apply to that sheet.




*Example validation codes in a template*

### Formatting reserved rows

In the majority of cases, rows 1-2 will not be visible to end users in plan files, due to freeze panes settings. If desired, you can also hide the rows in the template using standard spreadsheet functionality, to help prevent administrators from accidentally editing these rows.

The system-generated codes placed in these rows may be lengthy. Therefore, you may want to change the font size for these cells to 1pt, so that the cell contents do not impact any column auto-fit settings in the template.

## Column layout validation

If template validation is used on a sheet, then Axiom automatically validates the sheet column layout against the calc method library.

The column layout of a template sheet and its associated calc method library must remain in sync. For example, if you insert a column into a template sheet, but do not update the associated calc methods, then when those calc methods are placed into the sheet they will be "off" by one column. Depending on where the column was inserted, this may have a dramatic impact on your resulting plan files, such as breaking the save-to-database setup or other crucial template controls, and resulting in formula errors or unexpected formula results.

Additionally, you may have other templates in the file group with the same sheet name, which means those sheets use the same calc method library. If you make a column layout change to the sheet in one template, then you need to also make it in any other templates with the same sheet name. If you do not update all impacted templates, then the calc method library will be out of synch with the other templates.

If you need to make a change to the column layout of a template sheet, there is a standard procedure to follow. For more information, see Changing the column structure of a template sheet.

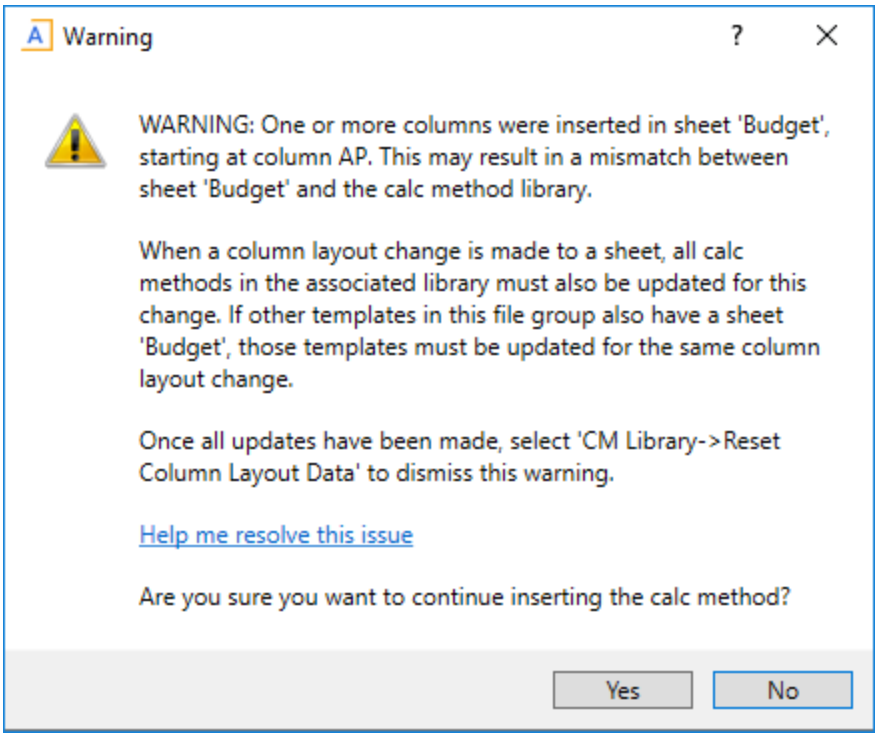
### Validation codes

The column layout validation is based on the system-managed validation codes that are placed in row 1 of the template sheet. Users should not place any other content in row 1, or edit any existing codes in row 1.

### Column layout warning

If Axiom detects that the column layout of the sheet has changed, the column layout warning will display. The warning is triggered based on the column layout change only—Axiom does not know whether you have made the associated updates to calc methods or other templates.

The warning displays each time that you save the template, or perform a calc method action in the sheet, until the column layout data is reset. Data should only be reset once you are confident that all necessary updates have been made.




*Example column layout warning*

# Resolving the column layout warning

## What causes this warning?

The column layout warning displays when Axiom detects a column structure change in a template sheet. You may have deleted columns in the sheet or added columns. The warning indicates the first detected change in the sheet (for example, "starting at column F") but there may be other column changes elsewhere in the sheet.

When a template sheet has an associated calc method library, the column structure of the sheet and the calc methods must remain in synch. Additionally, any other templates in the file group that contain the same sheet name must also have the same column structure, because they share the same calc method library.

This warning is intended to alert you of a possible mismatch between the template sheet, the calc method library, and other templates in the file group. Once triggered, the warning appears when you save the template (with the impacted sheet active), or when you perform any calc method action in the sheet.

## What do I need to do about this warning?

The action that you need to take depends on whether you are currently following standard procedures for updating the column structure of a template sheet. Generally, if you need to change the column structure of a template sheet, you need to bring in all calc methods *first*, then make the change, then update the calc methods in the library. For more information, see Changing the column structure of a template sheet.

- If you are currently following the standard procedure, then the warning is simply a reminder. Axiom only detects the column layout change; it does not know if you have already made all associated updates or if you are in the process of doing so. You can continue saving calc methods and templates until you have made all necessary updates. Once you are finished updating, you can "reset" the column layout data to dismiss the warning.
- If you get this warning and you are *not* following the standard procedure (for example, if you deleted or inserted a column without bringing in the calc methods first), then you most likely have a column mismatch that needs to be resolved. If possible, you should restore your template to its original state, and then follow the standard procedure for updating column layout. If this is not possible, or if you otherwise need assistance, please contact Axiom Support.

The warning will also occur if you manually delete a validation code in the reserved validation rows (rows 1-2). These rows are reserved for system-generated codes and should not be edited by users. If you absolutely know that this is the cause of the warning, then you can reset the column data layout to dismiss the warning and repopulate the validation codes. However, if you have any doubt, then you should review the template and the calc methods first to ensure that everything is still in synch.

### How do I reset the column layout data?

Once you are sure that all necessary updates have been made, you can reset the column layout data to dismiss the warning:

1. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Reset Column Layout Data**.

   **NOTE:** If other templates in the file group share the same sheet name, those templates will also be opened and the column layout data will also be reset for that sheet.

2. Save all templates where the column layout data was reset.

When you reset the column layout data, Axiom accepts the current column layout as correct, and places new validation codes in the sheet. You will not see the column layout warning for this sheet again, unless a new column layout change is made to the sheet.

**IMPORTANT:** You should not reset the column layout data unless you are confident that all necessary updates have been made. If a column mismatch is left unresolved, it can result in errors in your plan files, and may impact the validity of your data. If you have any doubt, please contact Axiom Support for assistance.

## Calc method identification and validation

If template validation is used on a sheet in a template or plan file, then Axiom automatically maintains a *calc method validation column* within that sheet. Whenever a calc method is placed into the sheet via any process, this column identifies:

- The name of the calc method.
- The starting and ending rows of the calc method.

The validation codes and calc method validation column will also be placed in driver files and file group utility files that use calc methods. The calc method features that depend on the calc method validation column can also be used in these files.

### Calc method validation column

When template validation codes are initially placed in row 1 of the sheet, the location of the calc method validation column is the last column in the used range (or column AD, whichever is greater). If the last column in the used range contains content, then the validation column is placed in the next column out (used range plus one).

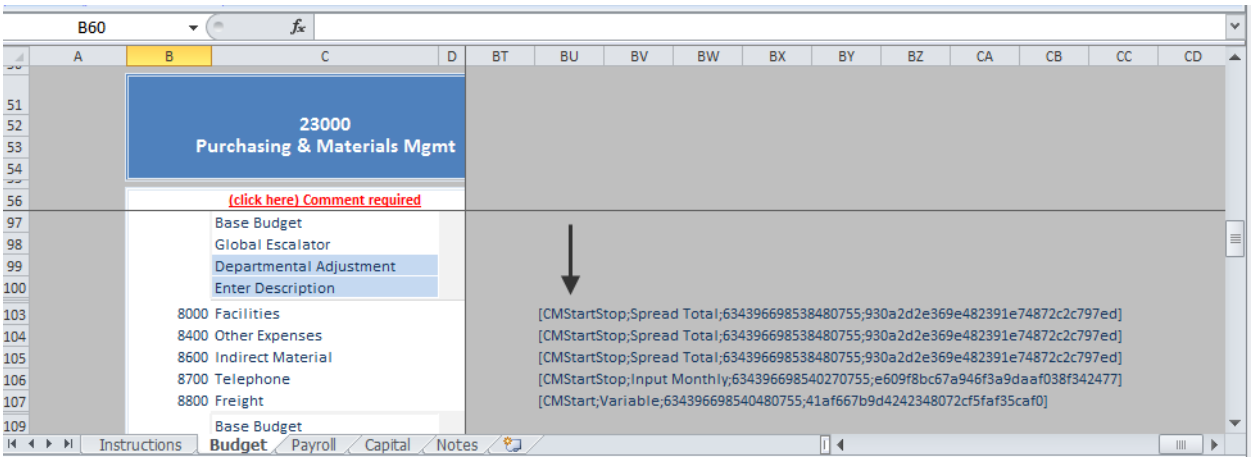
You can identify the calc method validation column by the text CM appended to the end of the template validation code. For example: `[AX;G;63432238403154596 6;CM]`.

As calc methods are inserted or changed in the sheet, calc method validation tags are placed in this column. The tags appear similar to the following:

```
[CMStartStop;Input Monthly;634322384641255966;97b5563fd4221cd]
```

This example is for a single-row calc method named Input Monthly. For multiple-row calc methods, there are separate `[CMStart]` and `[CMStop]` tags.

**IMPORTANT:** The tag syntax for calc method validation columns is mentioned here to help you identify the column and its tags in a sheet. The validation tags within this column are system-controlled and should not be manually edited.




*Example calc method validation column*

The calc method validation column is reserved for validation tags and should not contain any end user content. However, you can place reserved Axiom feature tags within this column. For example, you may want to hide this column as part of a view. In that case, you can enter the tag `[HideColumn]`.

The contents of the calc method validation column are validated when a file is saved. If Axiom detects content in this column that is not a calc method stop / start tag or an Axiom feature tag, a save error results. That content must be deleted before the save can continue. However, if a calc method stop / start tag has been modified so that the tag is invalid, the tag is simply ignored for any future calc method processes.

**NOTE:** Axiom may automatically move the location of this column, if it detects invalid content in the designated column when attempting to perform an action such as inserting a calc method. If this occurs, the contents of the column are moved, including existing calc method tags. Column formatting, if applicable, is not moved.

## Benefits of using calc method validation

The presence of the calc method validation column provides access to the following benefits:

- Administrators can replace multiple calc methods in a batch process, rather than needing to manually replace them one by one. See Replacing multiple calc methods.
- Administrators can use the Apply Calc Method Changes utility to update existing plan files for changes made to calc methods. See Updating plan files for calc method changes .
- When overwriting a multiple-row calc method, users no longer need to manually highlight all rows of the calc method. As long as the user's cursor is somewhere within the calc method, Axiom can automatically detect the starting and ending rows.
- Users can open calc method forms at any time by double-clicking the calc method in the sheet. Without the validation column, forms are only presented during initial insert or when using the calc method to overwrite another.
- When performing drill-down drilling on an Axiom query that uses a calc method library, the drill results will use the calc method applied to the row being drilled. The calc method validation column is used to detect the calc method for the current row.

6

# Calc Method Libraries

Calc method libraries store sets of predefined calculation methods. Each calc method is a set of formatting and formulas used to calculate a plan value. User inputs may be incorporated.

Calc method libraries can be used in the following ways:

- In Axiom queries, to dynamically build out rows of data. Each record to be inserted into the data range is assigned a specific calc method.
- By end users, to add new rows to plan files, or to change the calculations used on existing rows.

This section covers the administration of calc method libraries. For information on using calc methods to add or change rows in a plan file, see the *Desktop Client User Guide*. For information on using calc methods in Axiom queries, see the *Axiom File Setup Guide*.

## About calc method libraries

Calc method libraries are available in the following file group files: templates, plan files, drivers, and utilities. When working in other Axiom files, such as reports, you can use in-sheet calc methods. In-sheet calc methods apply to the current sheet only, are not saved back to a central library, and can only be used in conjunction with Axiom queries.

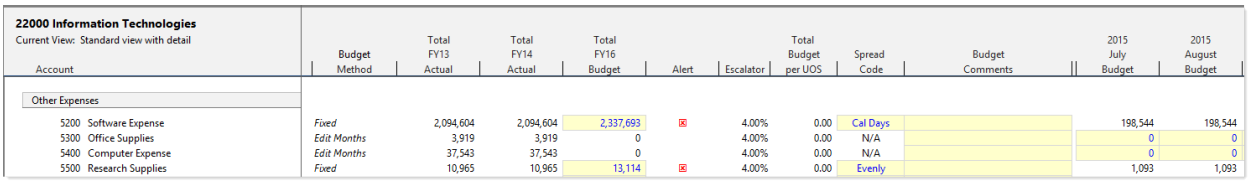
### What is a calc method?

Simply stated, a calc method is a row (or collection of rows) in a spreadsheet that has been saved back to a central location for re-use. The calc method can contain formatting, formulas, and values, but should be constructed so that it is adaptable for use in any relevant plan file.

When you save a new calc method, Axiom takes a copy of the selected row or rows and stores it in the associated calc method library file, along with a few settings relating to the calc method. When the calc method is used by an Axiom query, or a user chooses to use that calc method in a plan file, Axiom retrieves it from the library and inserts it into the sheet.

The contents of a calc method should be constructed so that it is applicable to all plan files where it may be used, and so that calc method change operations have the desired results. When a calc method is replaced with another calc method, Axiom compares the contents of the new calc method to the original calc method, and chooses what to overwrite depending on the type of contents. For more information, see Change calc method rules.

Each calc method can have a number of defined variables. When a user selects to insert or change a calc method, the user is prompted to define or select values for the variables, such as selecting an account number. When the calc method is inserted into the sheet, the selected variable values are placed into the designated cells of the calc method. For more information, see Defining calc method variables within the calc method properties (legacy approach).



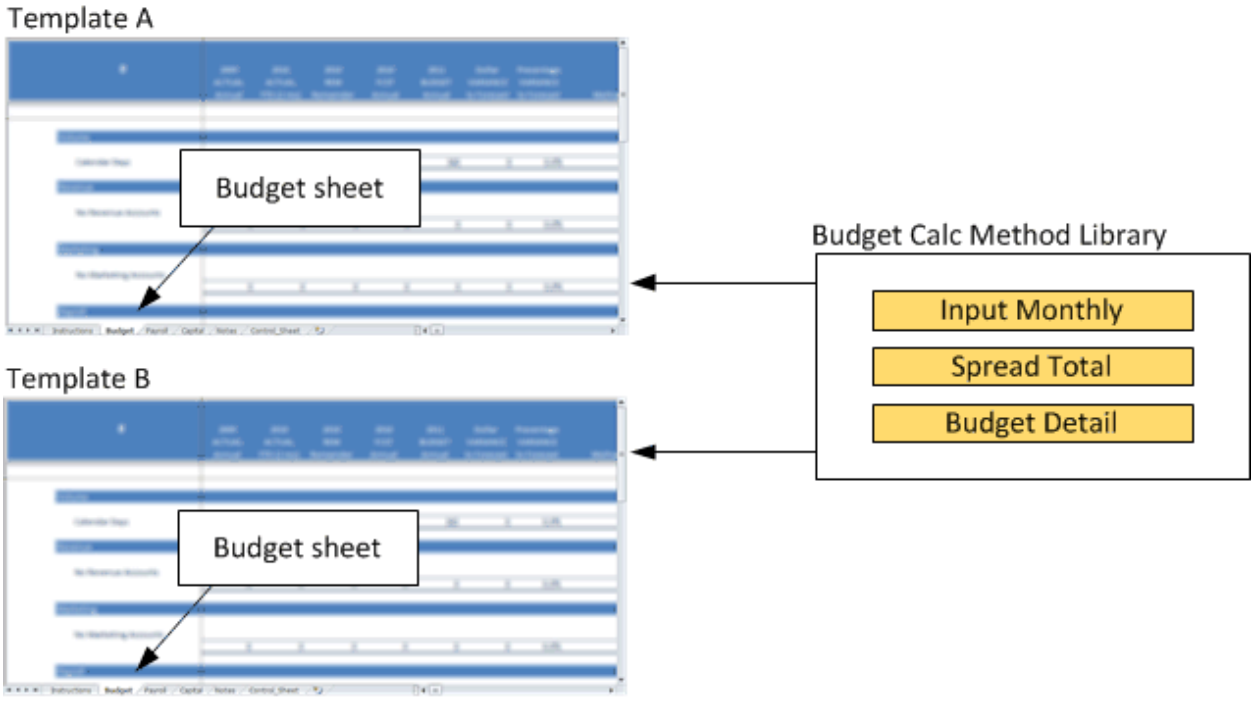
**22000 Information Technologies**
Current View: Standard view with detail

| Account | Budget Method | Total FY13 Actual | Total FY14 Actual | Total FY16 Budget | Alert | Escalator | Total Budget per UOS | Spread Code | Budget Comments | 2015 July Budget | 2015 August Budget |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Other Expenses | | | | | | | | | | | |
| 5200 Software Expense | *Fixed* | 2,094,604 | 2,094,604 | 2,337,693 | | 4.00% | 0.00 | Cal Days | | 198,544 | 198,544 |
| 5300 Office Supplies | *Edit Months* | 3,919 | 3,919 | 0 | | 4.00% | 0.00 | N/A | | 0 | 0 |
| 5400 Computer Expense | *Edit Months* | 37,543 | 37,543 | 0 | | 4.00% | 0.00 | N/A | | 0 | 0 |
| 5500 Research Supplies | *Fixed* | 10,965 | 10,965 | 13,114 | | 4.00% | 0.00 | Evenly | | 1,093 | 1,093 |

*Example plan file using calc methods*

## Calc method libraries and sheet names

Calc method libraries are stored based on sheet names, on a per file group basis. For example if you have a sheet in your template named Budget, then there is a calc method library named Budget, and all plan files in that file group that contain a Budget sheet can access those calc methods. If you have a second template in the same file group that also has a Budget sheet, that sheet uses the same Budget calc method library. Therefore, all plan sheets with the same name must also have the same column structure. If the sheet uses a different column structure, you must give it a different name.



### Users and calc methods

End users can use calc methods from a library to insert new rows or change existing rows in a plan sheet. Typically templates are constructed so that there are certain areas designed to accommodate the insertion of new rows. You can define custom insertion points to reinforce this structure and make it easy for users to insert new rows in the appropriate places, and you can use insertion controls to control exactly where calc methods can be inserted. See Setting up calc method controls for a sheet.

### Axiom queries and calc methods

Calc method libraries can be used in Axiom queries to build out rows of data as they are inserted into the sheet. When using this approach, each record to be inserted into the sheet must be assigned a calc method from the library. This can be done in the following ways:

- An assignment column can be created in the appropriate reference table, such as the ACCT table. Each account can be assigned a specific calc method. This column is then referenced in the Axiom query settings.
- Calc method assignments can be defined in a sheet within the file. This sheet and the applicable columns are then referenced in the Axiom query settings.
- A default calc method assignment can be defined in the Axiom query settings.

These three options can be used alone or together. For example, if all three options are defined, then Axiom first checks the sheet for an assignment, then the table column, and lastly uses the default assignment.

For more information on using calc methods with Axiom queries, see the *Axiom File Setup Guide*.

# Using calc methods

In order to manage calc method libraries, administrators often need to work with calc methods as end users would. For example:

- In order to edit the rows of an existing calc method, you must insert it into a sheet.
- When creating a new calc method, often an existing calc method is used as a starting point.
- Administrators should test calc method insertion and overwriting, to ensure that calc methods and calc method controls are set up as desired.

This section summarizes how to use existing calc methods, so that you can insert and change calc methods for the purposes of calc method development and testing. For more information on how end users work with calc methods, see the *Desktop Client User Guide*.

Administrators and users with the **Administer Calc Methods** permission can always insert or overwrite calc methods anywhere in a sheet, regardless of calc method controls. If you want to test the calc method controls set up for a template, you should log in using a non-admin user identity.

## Inserting a calc method into a sheet

When creating new calc methods, you should test the calc method insertion in several contexts, to ensure that the calc method is working as expected.

**To insert a calc method:**

1. Place your cursor in the row directly below where you want the calc method to be inserted.
2. On the **Axiom** tab, in the **File Options** group, click **Add Row(s) > Insert Calc Method(s)**.

   **NOTE:** You can also use the custom insertion points on the **Add Row(s)** menu to insert a calc method, if they have been set up. Keep in mind that when you use a custom insertion point, you will be limited to inserting the calc methods that have been enabled for that insertion point, even if you are an administrator. If you want to insert any calc method, regardless of calc method controls, then you must use **Insert Calc Method(s)** instead of a custom insertion point.

3. In the **Insert Calc Method** dialog, select the calc method that you want to insert, and then click **OK**.

   If you want to insert multiple instances of the calc method, edit the **Number of items to insert**. By default, this is set to 1.

   **NOTE:** If you are inserting the calc method for the purposes of editing the rows or creating a new calc method, clear the **Prompt for calc method variables** check box. This will keep the calc method "clean" for editing purposes. However, if you are testing the calc method, you should leave this check box selected, so that you can see how users will interact with the calc method variables. This option only displays on the dialog if the selected calc method uses variables.

4. If applicable, in the **Calc Method Variables** dialog, enter values for the calc method variables, and then click **OK**.

   This dialog only applies if the calc method has defined variables, and you left the **Prompt for calc method variables** check box selected. Otherwise, this dialog does not display and the calc method is simply inserted into the sheet.

## Changing an existing calc method

When creating new calc methods, you should test the change procedure for each calc method that will be enabled for overwriting (per the calc method controls in the sheet). You will want to ensure that each cell in the calc method overwrites (or doesn't overwrite) the existing contents as desired, per the calc method change rules. For more information on these rules, see Change calc method rules.

For example, if you allow an Input Monthly calc method to be overwritten by a Spread Total calc method (and vice versa), you should test this change procedure to make sure that existing inputs are handled as expected, formatting and formulas are applied as expected, etc.

**To overwrite an existing calc method with a new calc method:**

1. Place your cursor anywhere in the row (or rows) that you want to overwrite with a new calc method.

   If the original calc method in the sheet is a multiple-row calc method, it does not matter where you place your cursor, as long as it in within one of the rows of the calc method.

   **IMPORTANT:** This procedure assumes that your system has enabled template validation. Use of template validation allows Axiom to detect specific calc methods in the sheet, based on system-generated codes. If your system is not using template validation, then you must highlight all rows of the multiple-row calc method before you can overwrite it. For more information, see Template validation.

2. On the **Axiom** tab, in the **File Options** group, click **Add Row(s) > Change Calc Method**.
3. In the **Change Calc Method** dialog, select the calc method that you want to use to overwrite, and then click **OK**.

## Multiple-row calc methods and overwriting

You can replace single-row calc methods with multiple-row calc methods, and vice versa. The change operation intelligently deletes rows and inserts rows, depending on the row difference between the original calc method and the new calc method.

- If the new calc method has fewer rows than the original calc method, the additional rows in the sheet are deleted.
- If the new calc method has more rows than the original calc method, additional rows are inserted into the sheet.

For example, if you want to replace a single-row calc method with a five-row calc method, place your cursor in the row with the existing single-row calc method. The change operation replaces the single row with the first row of the five-row calc method, and then inserts four new rows to accommodate the rest of the five-row calc method.

If you want to replace a five-row calc method with a three-row calc method, place your cursor anywhere in the existing five-row calc method. The change operation replaces the first three rows of the five-row calc method with the new calc method, and then deletes the extra two rows in the sheet.

**IMPORTANT:** This procedure assumes that your system has enabled template validation. Use of template validation allows Axiom to detect specific calc methods in the sheet, based on system-generated codes. If your system is not using template validation, then you must highlight all rows of the multiple-row calc method before you can change it. In the previous example, you would need to manually highlight all five rows of the calc method within the sheet before changing it, because Axiom would be unable to automatically detect the starting and ending rows.

# Managing calc methods

You can add, edit, or delete the calc methods in a calc method library. These actions can be performed in any file that contains the sheet name corresponding to the library, within the appropriate file group. You must be in the sheet in order to manage the calc method library for that sheet.

Each calc method has two aspects that define the calc method:

- The row or rows of the calc method, as defined in a spreadsheet and saved to the calc method library.
- The calc method properties, such as the calc method name and description.

## Creating a new calc method

The easiest way to create a new calc method is to base it on an existing calc method. To do this, you can insert the existing calc method into the sheet, make the necessary edits, and then save it back as a new calc method.

The following procedure assumes you have already constructed the calc method inside the sheet, either by modifying an existing calc method, or by building a calc method from scratch.

**NOTES:**

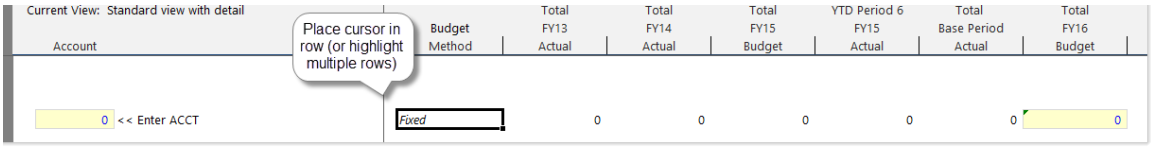
- When creating a new calc method, make sure that you have configured the contents of the calc method so that change operations will work as expected when a user uses the **Change Calc Method** feature. For example, if you want the contents of a specific cell to overwrite the existing contents every time, then you must set up that cell as a formula. For more information on how contents are replaced when changing a calc method, see Change calc method rules.
- Avoid using references to named ranges in calc methods. References to named ranges may unexpected errors in the workbook when calc methods are inserted using Office 365 but then viewed in older versions of Excel.

**TIP:** If you want to copy a calc method exactly, including items such as variables defined for the calc method, then you can duplicate an existing calc method. You can then make edits to the new calc method as needed. See Duplicating a calc method.

**To add a calc method to the library:**

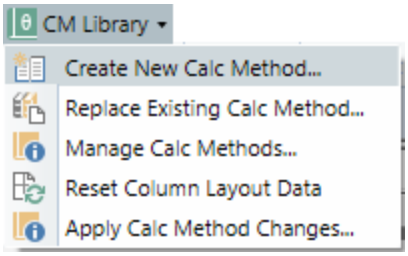
1. Place your cursor in the row that you want to save as a new calc method.

   If you are creating a multiple-row calc method, highlight all of the rows that you want to save as the calc method.




   **IMPORTANT:** Before saving the calc method, be sure that you have removed any department-specific data or inputs that you used to test the calc method. The calc method should not contain anything that you do not want to be inserted in all plan files where it can be used.

2. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Create New Calc Method**.




   **TIP:** You can also add a new calc method from the right-click menu (**Calc Method Library Admin > Create New Calc Method**).

   The **New Calc Method** dialog opens.

3. In the **Name** box, enter a name for the calc method.
4. Optional. In the **Description** box, enter a description for the calc method.

   It is recommended to enter a description. When users are selecting calc methods to insert, the description can help them choose the appropriate calc method.

5. Optional. In the **Group** box, do one of the following if you want to assign the calc method to a calc method group:

   - To define a new group, type the group name into the box. The group name will be saved when the calc method is saved.

- To assign the calc method to an existing group, select the group name from the drop-down list.

Calc method groups can be used with calc method controls, to specify the calc methods that can be used to insert or change at a particular location. If you specify a calc method group name in the `[InsertCM]` tag, then all calc methods that belong to the group can be used to insert or change at the location.

6. Optional. In the **Calc method variables** section, define one or more variables for the calc method.

   If calc method variables are defined, then a calc method form is presented to users whenever they use the calc method, to prompt them to define values for the variables. The variable values are then placed into the specified cells of the calc method.

   For more information on defining variables, see Defining calc method variables within the calc method properties (legacy approach).

7. Click **OK**.

The calc method is saved in the calc method library for the sheet name, for the current file group.

After saving the calc method to the library, delete the work rows from the sheet. The calc method is now stored in the library and does not need to be kept in the template.

## Calc method design considerations

When designing a new calc method, keep in mind that the calc method not only needs to account for the necessary user inputs and data calculations, but it must also be designed to work with any control columns and other work columns in the sheet. If a row inserted by a calc method needs to be flagged in some way—for example, to be included in a save-to-database—then the necessary tags must exist in the calc method itself so that those tags are present in the sheet when the row is inserted.

For example, consider the following:

- If a row in the calc method needs to be saved to the database, then a `[save]` tag (or the appropriate custom save tag) must be present in the save-to-database control column.
- If a row in the calc method needs to be hidden or adjusted by a view, the appropriate view tag must be present in the view control column.
- If a row in the calc method needs to be part of an action codes operation, the appropriate action tag must be present in the ActionCodes control column.
- If you want to allow users to change the current calc method, then an `[InsertCM]` tag must be present in the calc method control column (and appropriately configured to allow change).

This is not an exhaustive list of issues to consider; it is intended to illustrate the issue and get you thinking about all aspects of the calc method.

When a calc method is saved to the calc method library, only the cell-level formats are saved. If you have text in a cell that is formatted differently than the cell-level formats—for example, if only one word in the cell is formatted red or bold—that formatting will not be saved when the row is saved to the calc method library. When the calc method is retrieved from the library and inserted into a sheet, the cell-level formats are applied.

## Duplicating a calc method

You can duplicate an existing calc method to create a new calc method in the same library. Using this feature, you get an exact copy of the duplicated calc method, including the settings defined for the calc method, such as group assignments and calc method variables.

**To duplicate a calc method:**

1. Open a plan file or a template that uses the calc method library, and go to the sheet that corresponds to the library.
2. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Manage Calc Methods**.

   The **Manage Calc Methods** dialog opens, listing the calc methods for the current sheet. You can sort and filter this list using standard Axiom grid functionality.
3. Select the calc method that you want to duplicate, and then click **Duplicate**.

   The **Edit Calc Method** dialog opens, containing the settings for the duplicated calc method. By default, the name is "Copy of *CalcMethodName*."
4. Edit the name, description, group, or variables as desired, then click **OK**.

The new calc method displays in the **Manage Calc Methods** dialog. If the content of the calc method rows should be exactly the same as the original calc method, then you are finished. However, if you need to make edits to the calc method rows, then do the following:

- Insert the new calc method in a sheet: **File Options > Add Row(s) > Insert Calc Methods**.
- Edit the calc method rows within the sheet as desired.
- Place your cursor in the modified row (or select all of the rows, if it is a multiple-row calc method and you are not using template validation), and then replace the calc method: **Advanced > CM Library > Replace Existing Calc Method**.

## Editing calc method properties

All calc method properties are editable. If you change the name of an existing calc method, this may impact calc method assignments for Axiom queries or calc method controls. Make sure to update these areas as necessary after changing the name.

If you want to edit the calc method rows or rows, use **Replace Existing Calc Method** instead.

**To edit calc method properties:**

1. Open a plan file or a template that uses the calc method library, and go to the sheet that corresponds to the library.
2. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Manage Calc Methods**.

   The **Manage Calc Methods** dialog opens, listing the calc methods for the current sheet. You can sort and filter this list using standard Axiom grid functionality.

   > **TIP:** You can also access this dialog from the right-click context menu (**Calc Method Library Admin > Manage Calc Methods**).

3. Select the calc method that you want to edit, and then click **Edit**.
4. Edit the name, description, group, or variables as desired, then click **OK**.

The updated information for the calc method displays in the **Manage Calc Methods** dialog.

## Replacing a single calc method (editing calc method rows)

By replacing an existing calc method in a library, you can edit the row or rows of the calc method. If instead you want to edit the *properties* of a calc method, such as its name, description, or variables, then use **Manage Calc Methods**.

Keep in mind that when you replace the row or rows of a calc method, the changes only apply to new insertions of the calc method. Once a calc method has been placed in a sheet, it is not updated for future changes to the "master" calc method. If your plan files are already built out, and you need to make a change to a calc method that needs to be incorporated into all plan files, the best approach is to rebuild your plan files. If rebuilding plan files is not possible, you can use the **Apply Calc Method Changes** utility to update existing plan files for calc method changes. For more information on this utility, see Updating plan files for calc method changes .

> **NOTE:** If you need to replace multiple calc methods at a time—for example, after making a column structure change you need to update all calc methods in the library—you can use **Replace Multiple Calc Methods** instead. See Replacing multiple calc methods.

**To replace a calc method:**

1. Open a plan file or a template that uses the calc method library, and go to the sheet that corresponds to the library.
2. Insert the calc method that you want to update into the sheet (**File Options > Add Row(s) > Insert Calc Methods**).

**TIP:** If the calc method uses variables, then clear the **Prompt for calc method variables** check box in the **Insert Calc Method** dialog. This allows you to insert the calc method without defining variable values, so that it is "clean" for editing.

3. Make edits to the calc method in the sheet as desired.

   When you are finished making edits to the calc method, remove any department-specific data or inputs that you used to test the calc method. The calc method should not contain anything that you do not want to be inserted in all plan files where it can be used.

4. Place your cursor in the row containing the calc method. If this is a multiple-row calc method and your template is set up for template validation, then you can place your cursor anywhere in the calc method and it will automatically include all rows of the calc method. If you are not using template validation, or if you want to change the number of rows in the calc method, then you must highlight all applicable rows.

5. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Replace Existing Calc Method**.

   **TIP:** You can also access this dialog from the right-click context menu (**Calc Method Library Admin > Replace Existing Calc Method**).

6. In the **Update Calc Method** dialog, select the name of the calc method that you want to replace, then click **OK**. If your template is set up for template validation, then the name of the original calc method is already selected by default.

   A message box prompts you to confirm that you want to continue. So that you can be sure that the calc method is being saved with the appropriate number of rows, the message details the number of rows in the original calc method and in the edited calc method.

7. Click **OK** to continue.

## Replacing multiple calc methods

You can replace multiple calc methods in a library at one time. For example, if you want to insert a column in a template sheet, then you need to update all calc methods in the library for the new column structure. Instead of replacing each calc method one by one, you can update them all in a batch process.

**NOTE:** This feature is only available if you are using template validation. Use of template validation enables the calc method validation column, which allows Axiom to determine exactly which rows in the sheet are associated with each calc method. For more information, see Template validation and Calc method identification and validation.

**To replace multiple calc methods:**

1. Open a plan file or a template that uses the calc method library, and go to the sheet that corresponds to the library.

2. Insert the calc methods that you want to update into the sheet. On the **Axiom** tab, in the **File Options** group, select **Add Row(s) > Insert Calc Methods**.

    - In the **Insert Calc Method(s)** dialog, you can select multiple calc methods to insert. Note that if you select multiple calc methods to insert, calc method variables are ignored and no calc method forms will be displayed.
    - Multiple calc methods will be inserted into the sheet with a blank row between each calc method, so that it is easier to distinguish each individual calc method.
    - If **Include calc method name in column A** is selected, then the calc method name displays in the row above each calc method when the calc methods are inserted into the sheet, in the specified column (A is the default). The number of rows in the calc method is noted along with the name. This option is to assist in distinguishing each calc method while editing.

    **NOTE:** The ability to select multiple calc methods to insert is only available to administrators and users with the **Manage Calc Methods** permission. Other users can only select one calc method at a time.

3. Make edits to the sheet structure or the calc methods as desired.

    When you are finished making edits, remove any department-specific data or inputs that you used to test the calc methods. Calc methods should not contain anything that you do not want to be inserted in all plan files where it can be used.

4. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Replace Multiple Calc Methods**.

    The **Replace Multiple Calc Methods** dialog opens. This dialog lists all calc methods that are currently present in the sheet. For each calc method, the dialog lists:

    - The name of the calc method.
    - The number of rows of the calc method, and where that calc method is located in the sheet.
    - The date and time the calc method was inserted into the sheet.

5. Select the check boxes for the calc methods that you want to replace, and then click **OK**.

    You can select the check box in the header to select all calc methods.

    **NOTE:** If you have multiple instances of a particular calc method in the sheet, there is an entry for each instance. You can only select one of those instances to replace the master calc method. You can use the row location and the date/time of insertion to identify the instance that you want to use.

The selected calc methods are replaced in the calc method library.

If you are updating all calc methods as part of making a column structure change to the sheet, and now all relevant templates and calc methods have been updated, you should use **Advanced > CM Library > Reset Column Layout Data**. This will reset the data for the new structure and dismiss the column layout

warning. You should only reset the column layout if you are sure that you have updated all templates and calc methods that need to be updated. For more information, see Resolving the column layout warning.

## Deleting a calc method

You can delete a calc method from a library at any time. If you delete an existing calc method, this may impact calc method assignments for Axiom queries or calc method controls. Make sure to update these areas as necessary after deleting the calc method.

**To delete a calc method:**

1. Open a plan file or a template that uses the calc method library, and go to the sheet that corresponds to the library.
2. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Manage Calc Methods**.

   The **Manage Calc Methods** dialog opens, listing the calc methods for the current sheet. You can sort and filter this list using standard Axiom grid functionality.
3. Select the calc method that you want to delete, and then click **Delete**.

## Updating plan files for calc method changes

When calc methods are changed after plan files are already built out, those changes are not reflected in the plan files (unless the plan files are designed as rebuildable / virtual). If you want the change to apply to plan files, you must either rebuild plan files from the template, or you must use the **Apply Calc Method Changes** utility to update calc methods in existing plan files.

Using this utility, you can update the entire calc method in plan files, or you can update only a portion of the calc method. If you want to update portions of the calc method, you will need to specify ranges within the utility setup. It is a good idea to write down the applicable ranges before starting to configure the utility, so that you have the information at hand. For example, if you want to update columns T through Z of the first row of the calc method, this is noted as T1:Z1. If you want to update column T within all six rows of a six-row calc method, this is noted as T1:T6. You can then specify how you want the update to be applied for that portion—for example, change formulas and formatting, or formulas only, or formatting only.

If instead you choose to update the entire calc method, then the update behavior is the same as when using the **Change Calc Method** feature in plan files. In this case, you are changing from the old version of the calc method to the new version of the calc method. Make sure you are familiar with the change rules so that you understand how the update will occur (see Change calc method rules).

When you run the utility, you specify which calc methods to update (and which portions if applicable), and then you specify the plan files to process. You can run the utility in two modes:

- **Test mode**: You can select a single plan file to be updated on the client, so that you can test the update before processing multiple plan files. If the changes are not working as you intended, then you can simply not save the plan file.
- **Production mode**: Select multiple plan files to update, to be processed on the Axiom Scheduler Server. The plan files are saved as part of the processing.

**NOTE:** Your templates must be enabled for validation in order to use this utility. The utility depends on the calc method validation codes in order to perform updates in plan files. For more information, see Template validation.

**To update existing plan files for calc method changes:**

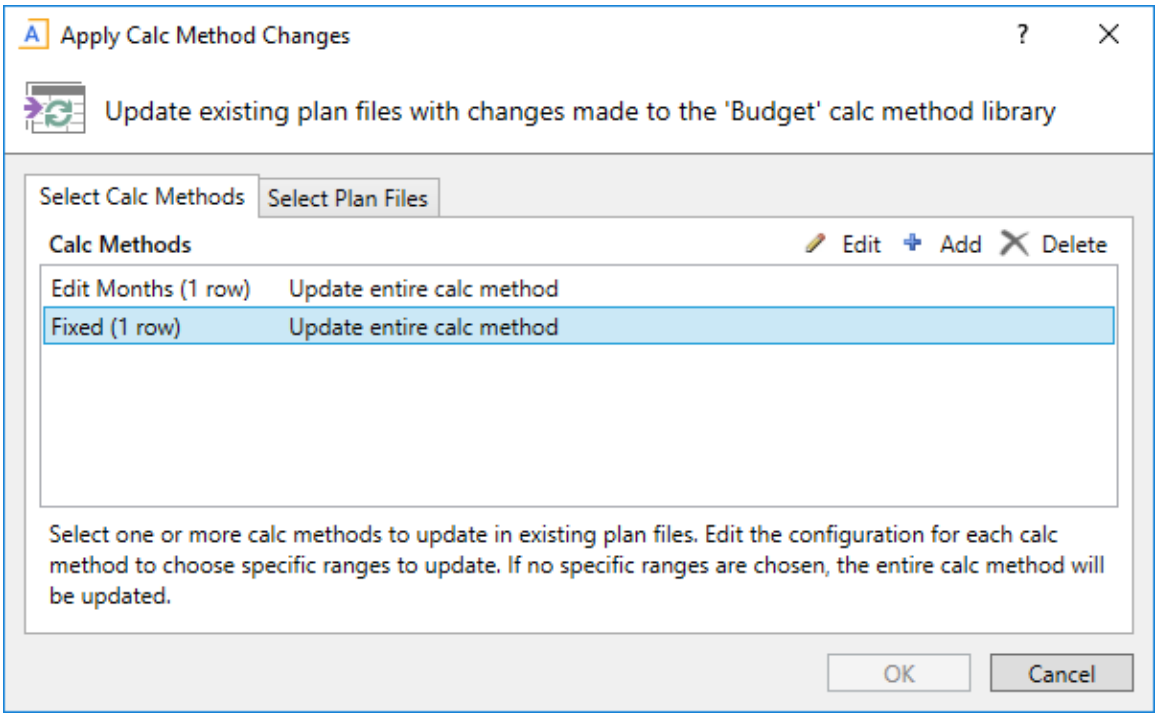
1. Open a template that contains the sheet name for the applicable calc method library, and go to that sheet.

   You can also start the utility from within a plan file, but in that case you will not be able to update the plan file that you have open.

2. On the **Axiom** tab, in the **Advanced** group, click **CM Library > Apply Calc Method Changes**.

   The **Apply Calc Method Changes** dialog opens. This dialog has two tabs. On the **Select Calc Method** tab you select calc methods to update in plan files. On the **Select Plan Files** tab, you specify which plan files to update.

3. On the **Select Calc Methods** tab, click **Add** to select a calc method to update. Repeat this process until you have added all calc methods that you want to update in this utility.




**NOTE:** If you previously configured this utility within the current Axiom session, you can load that prior configuration by clicking the **Load Previous Configuration** link in the top right corner of the dialog. If no previous configuration is available then this link will not be present.

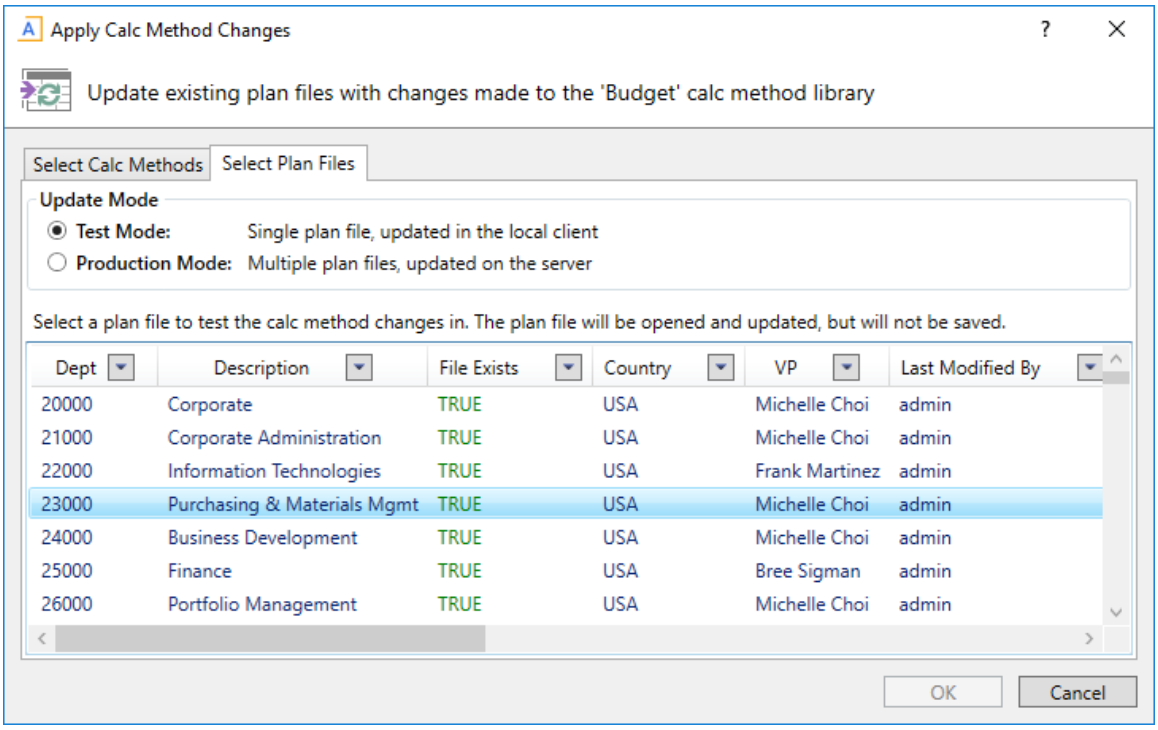
4. By default, the entire calc method will be updated in plan files. If instead you want to specify a particular portion of the calc method to update, select the calc method in the list and then click **Edit**.

   For more information on how to specify a portion of the calc method to update, see Changing only specified portions of a calc method.

5. On the **Select Plan Files** tab, select either **Test Mode** or **Production Mode** as the update mode. Then, specify the plan file(s) to update:

   - If you selected **Test Mode**, then select a single plan file to test the updates. For more information, see Test mode.
   - If you selected **Production Mode**, then select all plan files that you want to update. The selection process is similar to the Process Plan Files utility—you can select individual plan files in the list, or you can use a filter to specify plan files, or you can process all plan files. For more information, see Production mode.

   In this example, test mode is selected, so the changes will be applied to the selected plan file (Dept 23000).




6. Click **OK** to begin processing. A message box displays the number of plan files you are about to update, and asks you to confirm that you want to continue. If you are using production mode, then the message also informs you that a restore point will be created before the process begins, so that if necessary you can restore any plan files to the state they were in before running the utility (see Restoring plan files from restore points).
7. Click **Yes** to continue. If the number of plan files is not as expected, you can click **No** to cancel the process and return to the **Apply Calc Method Changes** dialog.

The process begins, using either test mode or production mode.

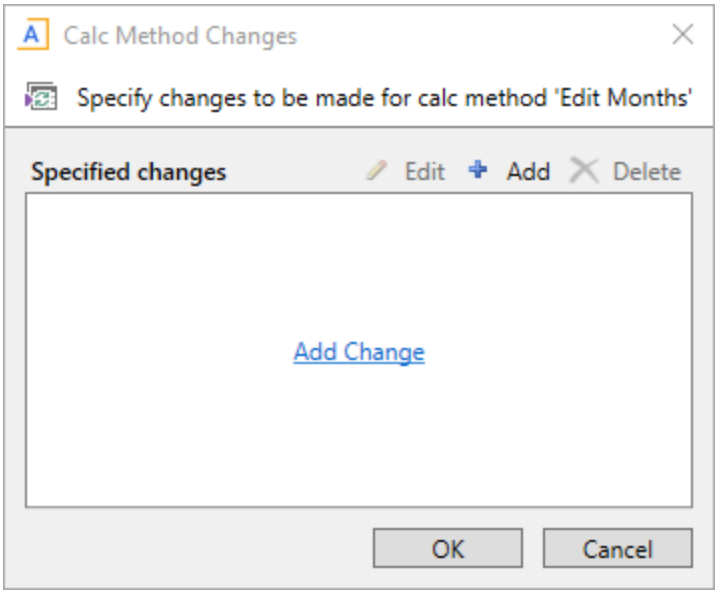
**NOTE:** If you use the utility to update a calc method that is associated with a master sheet, the calc method will be updated in all copied sheets that use the same calc method library as the master sheet.

## Changing only specified portions of a calc method

By default, if a calc method is listed in the **Select Calc Methods** tab, then the utility will update the entire calc method in plan files. Alternatively, you can specify a particular portion of the calc method to update, leaving the rest of the existing calc method unchanged.

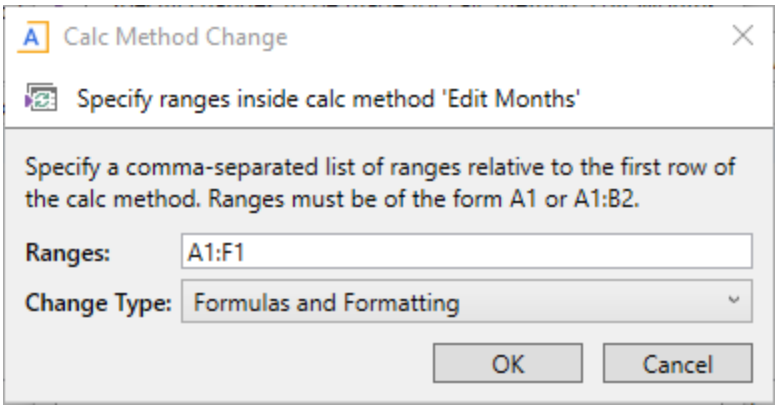
1. Select the calc method in the **Select Calc Methods** tab, and then click **Edit**.

2. In the **Calc Method Changes** dialog, click **Add** to specify the change to be made to this calc method.

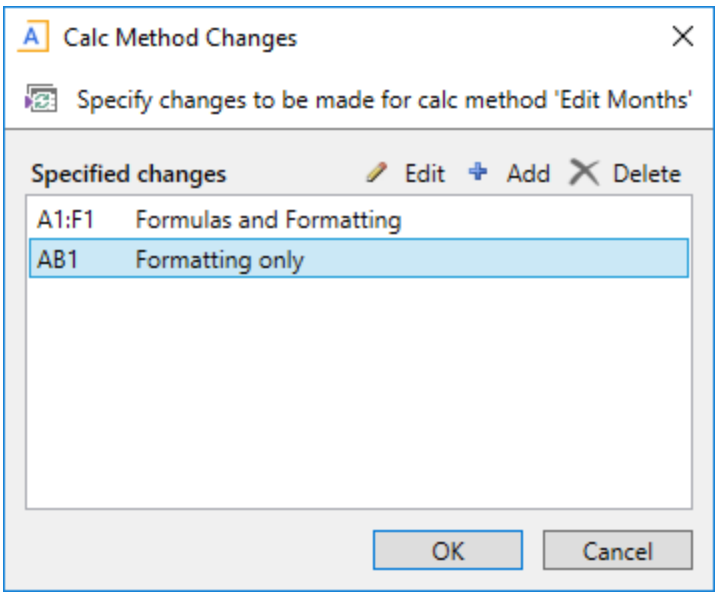



3. In the **Calc Method Change** dialog, define the range to update and the change type, and then click **OK**.

| Item | Description |
|---|---|
| Ranges | Specify one or more ranges to update within the calc method. Separate multiple ranges with commas.<br><br>Ranges are relative to the first row of the calc method. For example, if you want to update columns A through F of the first row of the calc method, this is noted as A1:F1. If you want to update column F within all six rows of a six-row calc method, this is noted as F1:F6. |
| Change Type | Specify the change type as either **Formulas and Formatting** (default), **Formulas only**, or **Formatting only**. |



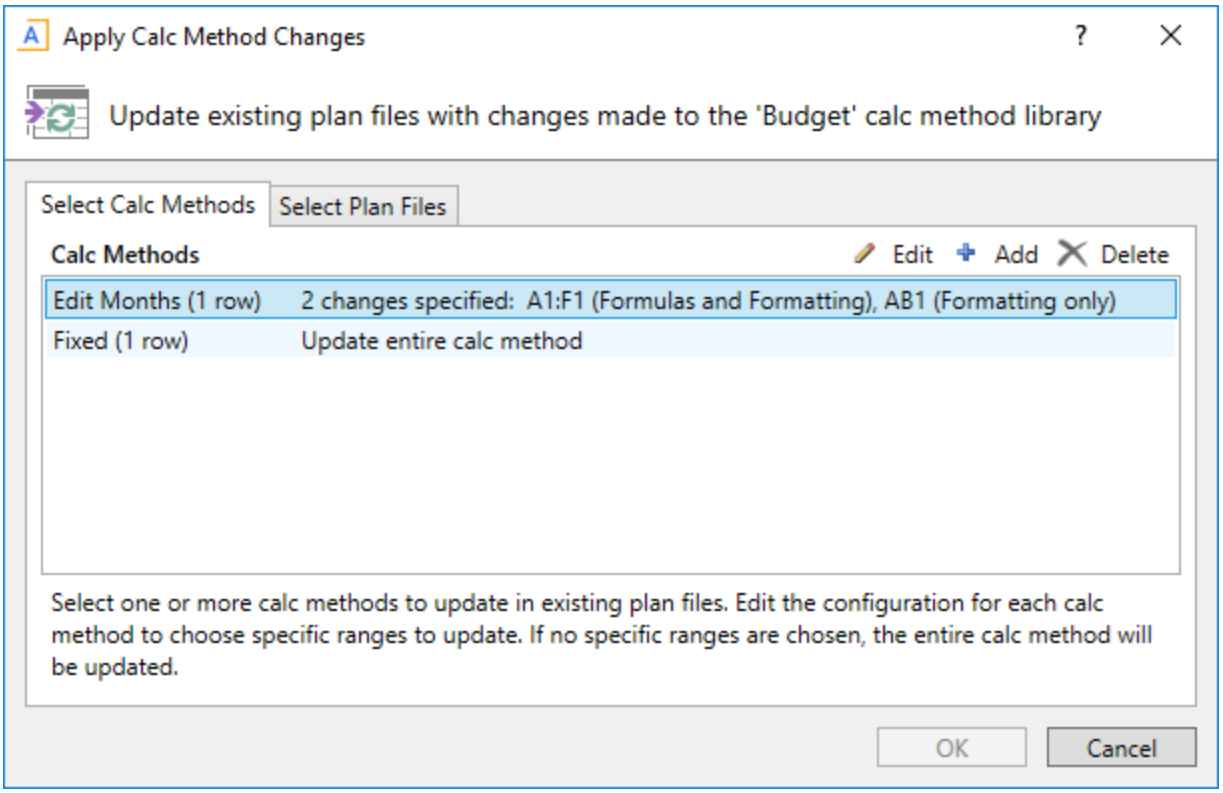
4. Repeat steps 2 and 3 until you have added all desired changes for the calc method. You can add as many changes as desired. After you have defined one or more changes, the Calc Method Changes dialog looks like the following example screenshot.




   - If you need to edit a range or a change type, select the change and then click **Edit**.
   - If you need to delete a change, select the change and then click **Delete**.

5. When you are finished adding changes, click **OK** to close the **Calc Method Changes** dialog and return to the **Select Calc Methods** tab.

Your specific changes to the calc method are now noted in the dialog, as shown in the following example:



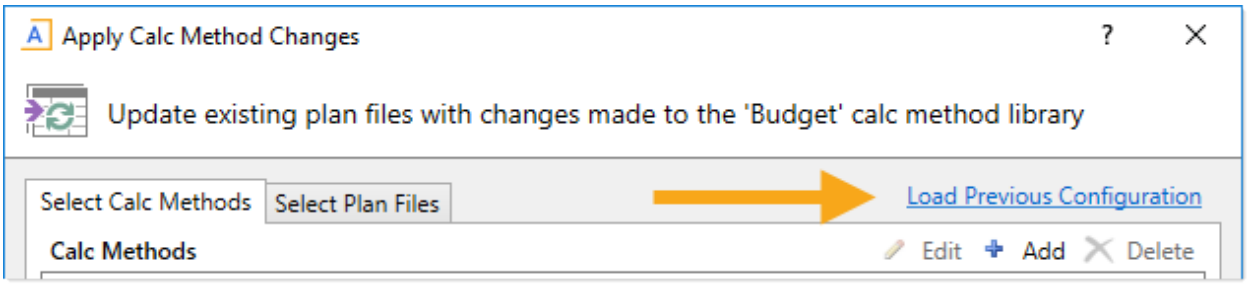

## Test mode

When running the utility in test mode, the selected plan file is opened in the client and updated. You can then check the updates to be sure they were applied as expected.

If the test was successful then you can save the plan file, or you can close it without saving and then include it when you run the utility in production mode.

If the test was not successful, then you can make further changes and test the utility again. The issue may be with the calc method itself (perhaps the modifications to the calc method were not correct) or with the utility configuration (perhaps the range to update was incorrect, or the change type was incorrect).

After running the utility in test mode, you want to be able to access the utility configuration again to re-test it or to run it in production mode. Your previous configuration will be saved for the duration of the current Axiom session. The next time you enter **CM Library > Apply Calc Method Changes**, you will have the option to **Load Previous Configuration** instead of starting from scratch.




This option will be available until you exit Axiom; at that point the utility configuration will be cleared.

### Production mode

When running the utility in production mode, the selected plan files are processed by the Axiom Scheduler Server. Each plan file is updated according to the utility configuration, and then the plan file is saved.

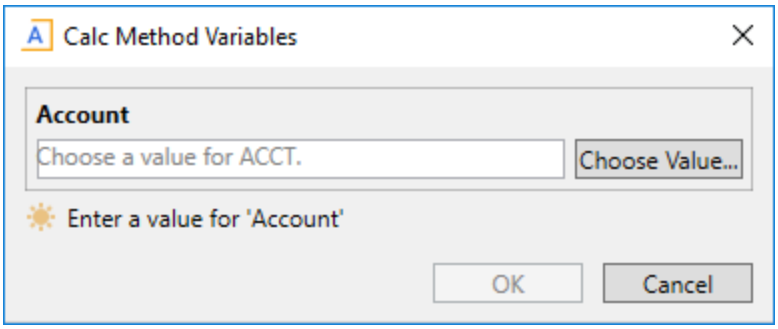
**NOTE:** A save-to-database is not performed as part of this utility. If the calc method changes resulted in data changes that you want to save to the database, then you should run Process Plan Files once the utility is finished.

A notification will display in the Notifications task pane when the processing is complete. If any errors occurred during processing, this will be detailed in the notification.

# Using calc method variables

You can define one or more variables for a calc method in a calc method library, so that users are prompted to specify values for the variables when using the calc method. When a user inserts the calc method into a sheet, or uses the calc method to overwrite another calc method in the sheet, Axiom first displays a dialog that prompts the user to specify values for the variables. After the user specifies the values, the insert or overwrite operation continues, and the values are placed in the appropriate cells.

For example, many calc methods require the user to specify an account code. Instead of making the user identify the appropriate cell in the sheet and then enter or select an account code after the calc method has been inserted, a calc method variable can be used to prompt the user to select the account code as part of the insertion process.




*Example calc method variables in the Desktop Client*

When you define a calc method variable, you specify properties such as the following:

- The name of the variable, to be displayed on the calc method form to identify what the user is entering or selecting.
- The type of variable—for example, whether the user is manually entering text or a number, or whether the user is selecting from a predefined list.
- The relative location of the variable, to determine where the specified value is placed in the calc method.
- Whether the user is required to specify a value for the variable.
- Whether the variable only displays when the calc method is initially inserted.

All defined variables for the calc method are displayed in the same dialog. You can specify the order in which the variables are presented on the form.

Calc method variables apply when users insert or change (overwrite) calc methods in a plan file. Variables are ignored when calc methods are inserted via an Axiom query.

**NOTE:** The normal change rules do not apply to calc method variables. If the new calc method has a variable defined, the variable value specified by the user always overrides the existing cell contents. However, if the user does not specify a variable value (if the variable is not required and the user leaves the entry blank, or if the variable is excluded from the change operation because it is set to **Insert Only**), then the normal change rules apply to that cell.

If a plan file is form-enabled, then users will also be prompted to select variable values when they insert calc methods using the Add Rows command. For more information, see the *Axiom Forms and Dashboards Guide*. Axiom forms do not currently support changing calc methods on existing rows, so calc method insertion is the only opportunity for users to specify variable values in this context.

### Options to define calc method variables

There are two ways to define calc method variables:

- **Using a data source in the spreadsheet (Preferred)**: You can define calc method variables using a CalcMethodVariables data source within a sheet of the spreadsheet. This feature takes more effort to set up, but supports more variable types and features.
- **Within the calc method properties (Legacy)**: You can define calc method variables directly within the calc method properties. This feature is simpler to set up, but supports fewer variable types and features.

The data source feature is the preferred approach to defining calc methods, and is intended to be the focus of any enhancements moving forward. The calc method properties feature is considered to be the legacy approach, and is supported primarily for backward-compatibility. Although there are currently no plans to remove the legacy approach, we strongly encourage adopting the new approach for any new calc methods, or as part of any significant design changes to existing calc methods.

## Defining calc method variables using a CalcMethodVariables data source

You can define calc method variables using a CalcMethodVariables data source. This provides a dynamic method of defining calc method variables, because formulas can be used in the sheet to enable or disable variables, or to change variable configuration.

In order to use a CalcMethodVariables data source with a calc method, you must:

- Enable **Variables Use Data Source** for the calc method. This can be done when creating the calc method or editing the calc method properties.
- Define a CalcMethodVariables data source in the file where calc methods will be inserted.

### How the data source works

When a user inserts or changes a calc method, and **Variables Use Data Source** is enabled for the calc method, then Axiom checks the file for the presence of a CalcMethodVariables data source with a matching sheet name and calc method name. For example, if the calc method library is for sheet Budget and the calc method is named Insert Only, then Axiom looks for the following tag: `[CalcMethodVariables;Budget;Insert Only]`.

If a matching data source is found, then it is used to present calc method variables to the user. If a matching data source is not found, then the calc method is inserted without variables. The lack of a matching data source is not an error condition.

Because the data source is defined in the file and evaluated at the time of calc method insertion, it can use formulas to change dynamically. You can dynamically enable or disable certain variables (or the entire data source), and you can change the configuration of variables. For example, formulas can be used to determine the filter to apply to a Grid or ComboBox variable, to filter the list of available column values.

The CalcMethodVariables data source is very similar to the RefreshVariables data source, and supports the same set of variables. However, note the following differences:

- **Default value**: Calc method variables do not support defining default values. The user must specify a value for the variable, or else it is treated as not defined. If you do not want users to leave the variable blank, then it should be configured as required.
- **Selected value**: When the calc method is inserted into the sheet, the user's selected values for the variables are not placed in the `[SelectedValue]` column of the data source. Instead, they are placed in a designated location within the calc method, as defined by the `[RelativeLocation]` property.

  However, if you have configured a variable as a dependent variable, the `[SelectedValue]` column is still used to temporarily write back the selected value of the parent variable. This is done so that formulas can resolve and the dependent variable can update based on the selected value for the parent variable. This process happens in the background during the calc method insertion, and does not affect the file.

Multiple files in a file group can use the same calc method library, if those file share the same sheet name. When using a data source to define calc method variables, the data source must be defined in each file that uses the library, because the variables are not stored in a centralized location. For example, if you have two templates with a Budget sheet that use the Budget calc method library, each template must contain a copy of the data source. This could potentially be an advantage, if you want to use a different variable configuration in each template. However, if both templates use the same variable configuration, then you must set up and maintain the data source in both templates. In this case, if you do not need the additional variable types or features provided by the data source, you may find it easier to use the legacy variables in the calc method properties.

**NOTES:**

- If a CalcMethodVariables data source contains dependent variables, and the calc method insertion occurs in the Desktop Client, then the `[SelectedValue]` column does not get cleared after the insertion occurs. This is a known issue that is intended to be fixed in a later patch or release. It is not intended to persist values in the `[SelectedValue]` column.
- If you are inserting a calc method in an Axiom form, and the associated CalcMethodVariables data source has a value in the `[SelectedValue]` column, that value will be populated into the Calc Method Variables dialog (but only on initial insertion of the calc method). This is a known issue that is intended to be fixed in a later patch or release. It is not intended to support default values in the CalcMethodVariables data source.

## Defining the CalcMethodVariables data source

The CalcMethodVariables data source must be defined in the file where calc methods will be inserted. For templates/plan files, the data source is defined in the template used to create the plan files.

The data source can be defined on any non-control sheet of the file. In practice, the data source is typically placed on a designated sheet named something like Variables. If the file is a standard Axiom spreadsheet file that users will access in the Excel Client or Windows Client, then this sheet is typically hidden from end users (though it does not have to be).

**To create a CalcMethodVariables data source:**

- Right-click the cell in which you want to start the data source, then select **Axiom Wizards > Insert Calc Method Variable Data Source**.

  This option is only available in files that belong to a file group. Only file groups can use calc method libraries.

The wizard adds the primary tag, all column tags, and one row tag to define a single variable. To create variables, you can manually populate the variable properties.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

The following screenshot shows an example CalcMethodVariables data source with two variables:

| | A | B | C | D | E | F | G | H | I | K | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 27 | | | | | | | | | | | |
| 28 | | | [CalcMethodVariables;Budget;Edit Months] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [DependsOn] | [ColumnName] |
| 29 | | | [Variable] | Acct | Account | ComboBox | R1 | TRUE | TRUE | | acct.acct |
| 30 | | | [Variable] | Desc | | RelatedColumnValue | S1 | | TRUE | Acct | acct.description |

The CalcMethodVariables data source uses the following syntax:

### Primary tag

**[CalcMethodVariables;*SheetName*;*CalcMethodName*]**

The *SheetName* identifies the sheet associated with the calc method library. This is the sheet where the calc method is inserted.

The *CalcMethodName* identifies the calc method for the variables. This is the name of the calc method as defined in the calc method properties.

For example, if the calc method is named Edit Months and the calc method library is for the Budget sheet, then the primary tag should be renamed as follows:

```
[CalcMethodVariables;Budget;Edit Months]
```

The placement of this primary tag defines the control column and the control row for the data source.

- All column tags must be placed in this row, to the right of the tag.
- All row tags must be placed in this column, below the tag.

### Row tags

**[Variable]**

Each row flagged with this tag defines a unique calc method variable. Generally speaking, each variable represents a value that you want the user to input or select.

### Column tags

Each column in the data source defines a variable property, such as the variable name and type, and whether the variable is enabled. All variables share a set of general properties. Additionally, certain variables have additional properties that only apply to that particular variable type.

**[VariableType]**

This column defines the variable type. The variable type determines the valid property columns for the variable.

The following variable types are most commonly used for calc method variables:

- **Calendar**: The user can select a date from a calendar.
- **ComboBox**: The user can select a value from a drop-down list. The list can be generated based on a specified table column, or an Axiom query, or a ComboBox data source. The user can type into the box to filter the items in the list.
- **Decimal**: The user can enter any numeric value, including decimals.
- **Grid**: The user can select a value from a specified table column (for example, ACCT.ACCT to select from a list of accounts). The column values are presented in a searchable grid dialog. Multi-select can be enabled.

- **GUID**: This is a special variable type that is not presented to the user. Instead, a globally unique identifier is automatically generated and placed in the calc method.
- **Integer**: The user can enter any whole number.
- **RadioButton**: The user can select a single value from among two or more radio buttons. The list of radio buttons can be generated based on a specified table column, or an Axiom query, or a ComboBox data source.
- **RelatedColumnValue**: This is a special variable type that is not presented to the user. Instead, it is used to return a related column value for a parent variable, and place that value in the calc method. For example, if the user is selecting an account code, this can be used to place the account description in the calc method.
- **String**: The user can enter any string value.
- **StringList**: The user can select any item in a manually defined list.

The following additional variable types are technically supported for use in calc method variables, but typically they are not useful in this context. If you want to use one of these variables, see the variable information as documented for refresh variables, in the *Axiom File Setup Guide*.

- **AdvancedFilter**: The user can create a filter criteria statement using the Advanced Filter view, or create a limit query statement.
- **CheckBox**: The user can select or clear a check box.
- **HierarchyFilter**: The user can select one or more values from a defined hierarchy to result in a filter criteria statement. Note the following limitations when using this as a calc method variable as opposed to a refresh variable:
  - The property `[UseAsQuickFilter]` does not apply to calc method variables and is not present in the CalcMethodVariables data source.
  - This variable does not currently work in the Web Client when used as a calc method variable.
- **RangeSlider**: The user can select the top and bottom values within a defined range, using slider buttons.
- **Slider**: The user can select a single value within a defined range, using a slider button.

Note the following about placement of the data source tags:

- Column tags can be in any order. Optional column tags can be omitted from the data source if they are not being used. For example, if none of your variables are StringList type, you can omit the `[ListChoices]` tag.
- Column and row tags do not have to be continuous. Axiom will continue searching the control row and control column for valid tags until it reaches a bracketed tag that does not belong to the data source.

### Testing calc method variables

When the file is saved, certain calc method variable settings in the data source are validated and will cause an error if invalid. After completing the variable set up, you should save the file and correct any errors found.

**NOTE:** Axiom does not validate the sheet name and the calc method variable name in the primary CalcMethodVariables tag. If your variables are not being recognized when the calc method is inserted, you should make sure that **Variables Use Data Source** is enabled in the calc method properties, and verify that the correct sheet name and calc method name are used in the primary data source tag.

Once you have set up the file as desired and corrected any invalid settings, you should test the calc method variables by inserting the calc method into the sheet, using the same method that you intend end users to use. Make sure that:

- The variable name and selections make sense from a user perspective. If it does not seem clear what you are asking the user to do, you may want to edit the variable name or use a different variable type.
- If the variable is not required, test a blank entry to make sure that the calc method still works as expected if no value is specified. If not, you may want to edit the calc method design or make the variable required.
- If you are using dependent variables, make sure that any dynamic settings are working as expected. Try entering different values for the parent variable to make sure you have accounted for all possibilities.

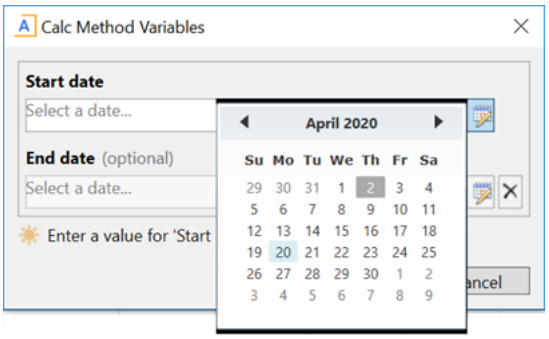
## Calendar calc method variable

Calendar calc method variables prompt users to select a date from a calendar. The date is written back to the specified location in the calc method.

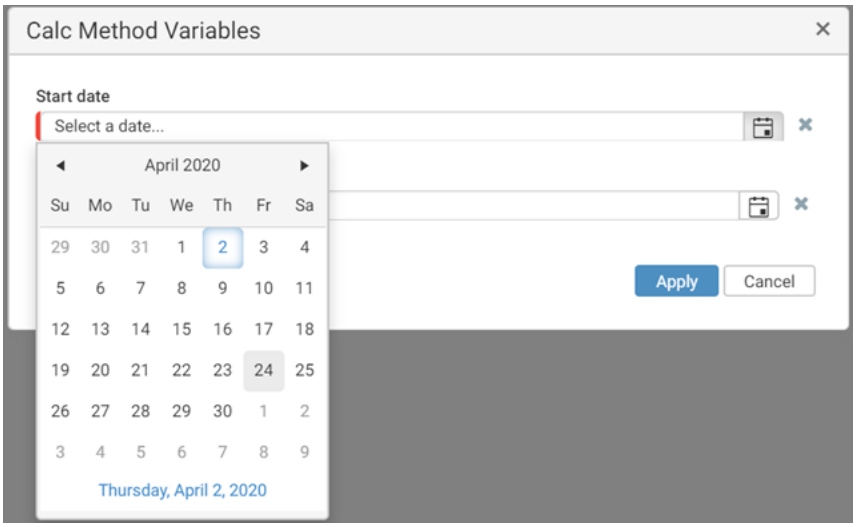
When used in the Web Client, calendar variables can also be used to select a month or year.

### Variable behavior

The variable displays as a text box with a calendar button next to it. The user can click the button to select a value from the calendar.




*Desktop Client: Example Calendar calc method variable*

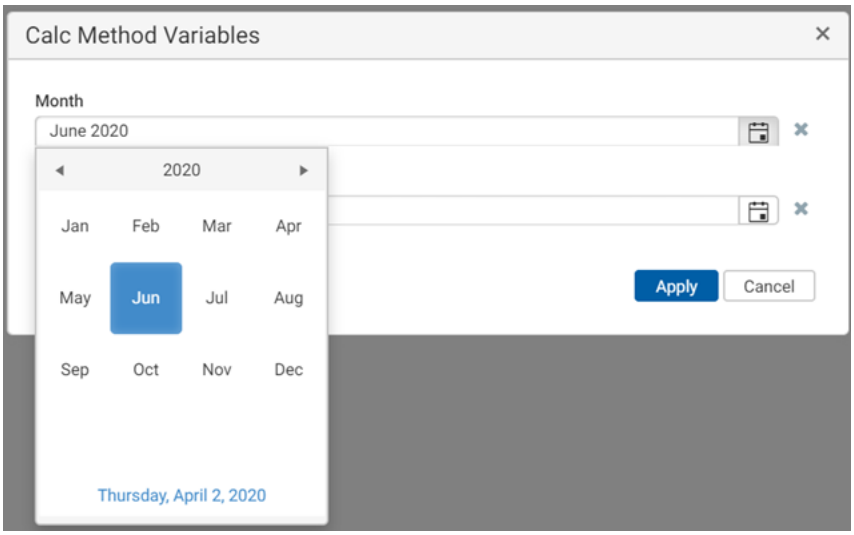



*Web Client: Example Calendar calc method variable*

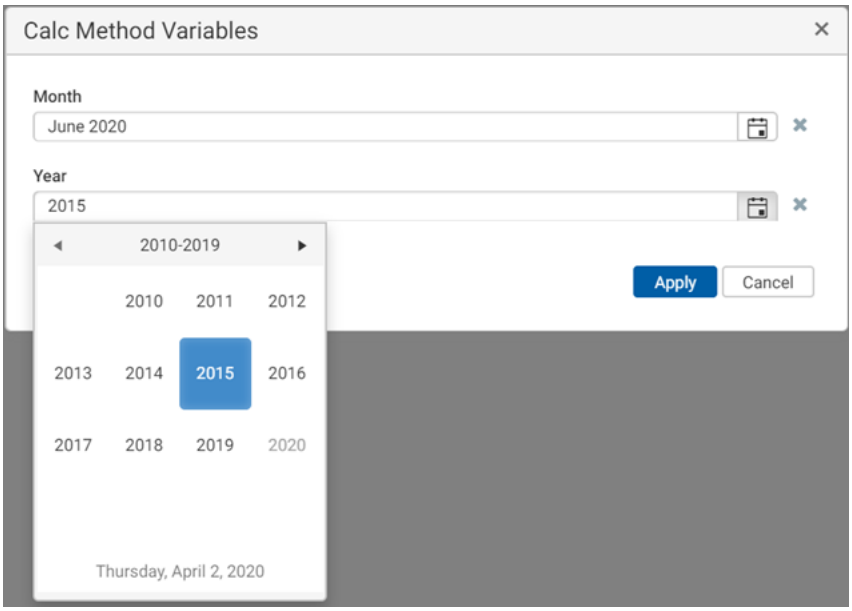
In the Web Client, the current date shows at the bottom of the calendar for reference.

Once the user has selected a date from the calendar, the selected date displays in the text box. The user cannot type a date into the text box; only selection from the calendar is allowed.

In the Web Client only, Calendar calc method variables can also be used to select a month/year combination, or a year. The month and year options are not available in the Desktop Client.




*Web Client: Example Calendar calc method variable to select month/year*




*Web Client: Example Calendar calc method variable to select year*

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a Calendar variable. Some data source columns do not apply in this case and are not discussed here. If these inapplicable columns are present in the data source, they should be left blank on rows that define Calendar variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties

The following additional properties apply to Calendar variable types:

| Column Tag | Description |
|---|---|
| `[MinDate]` | Optional. The earliest date that is valid for a user to select in the calendar. If specified, the calendar control will not allow the user to select a date that is earlier than this date.<br><br>When using the Month or Year display format, the minimum date must still be a full date. The appropriate minimum month and year will be determined from that date.<br><br>**NOTE:** The cell format for this property should be set to Date. DateTime formats are not supported and may cause errors. For more information, see Handling date formats. |
| `[MaxDate]` | Optional. The latest date that is valid for a user to select in the calendar. If specified, the calendar control will not allow the user to select a date that is later than this date.<br><br>When using the Month or Year display format, the minimum date must still be a full date. The appropriate minimum month and year will be determined from that date.<br><br>**NOTE:** The cell format for this property should be set to Date. DateTime formats are not supported and may cause errors. For more information, see Handling date formats. |
| `[DisplayFormat]` | Optional, applies to Web Client only. Specifies the type of date value for selection:<br>• **Date**: Users select specific dates from a calendar control. This is also the default behavior if no display format is specified.<br>• **Month**: Users select a month and year combination from a drop-down selection.<br>• **Year**: Users select a year from a drop-down selection.<br><br>The display format determines the values for selection and the display of the selected value in the variable. However, the return value written to the `[SelectedValue]` field is always a full date. See Handling date formats for more information. |

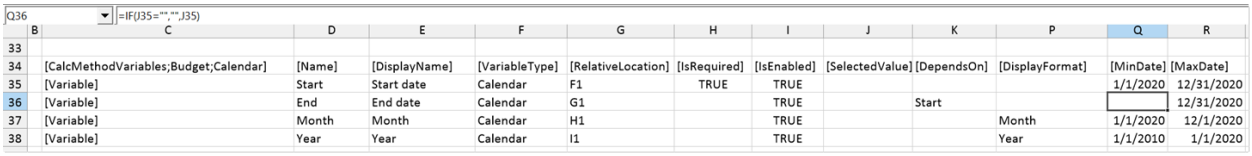
| Column Tag | Description |
|---|---|
| [AutoQuoteString] | Optional. Specifies whether the date value is placed in single quotation marks when it is written to the [SelectedValue] column (True/False). If omitted or blank, the default setting is False, which means the date value is not quoted. |

The following properties do not apply to Calendar variables: PlaceHolderText, ListChoices, ColumnName, AdditionalColumns, ColumnFilter, AllowMultiSelect, DataSourceName, Hierarchies, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Example data source

The following screenshot shows example Calendar variables.

- The first two variables define start and end dates. The End variable is dependent on the Start variable, so that the minimum value of the end date can use a formula that points to the selected value of the start date. This is to ensure that the user does not accidentally set the end date to an earlier value than the start date. Note that currently this configuration only works in the Desktop Client.
- The third and fourth variables use the [DisplayFormat] to select a month/year combination and a year. This configuration is only valid for use in the Web Client; it will be ignored in the Desktop Client.



Q36 =IF(J35="","",J35)

| | B | C | D | E | F | G | H | I | J | K | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | | | | | | | | | | | | | |
| 34 | | [CalcMethodVariables;Budget;Calendar] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [SelectedValue] | [DependsOn] | [DisplayFormat] | [MinDate] | [MaxDate] |
| 35 | | [Variable] | Start | Start date | Calendar | F1 | TRUE | TRUE | | | | 1/1/2020 | 12/31/2020 |
| 36 | | [Variable] | End | End date | Calendar | G1 | | TRUE | | Start | | | 12/31/2020 |
| 37 | | [Variable] | Month | Month | Calendar | H1 | | TRUE | | | Month | 1/1/2020 | 12/1/2020 |
| 38 | | [Variable] | Year | Year | Calendar | I1 | | TRUE | | | Year | 1/1/2010 | 1/1/2020 |

## Handling date formats

Within the Calendar calc method variable, the selected values are displayed as follows, depending on the specified [DisplayFormat]. The exact format depends on your system locale.

- **Date**: Dates are displayed in an Excel "short date" format, such as 1/1/2021.
- **Month**: Months are displayed in month/year format, such as January 2021. Only applies to Web Client.
- **Year**: Years are displayed as the year number, such as 2021. Only applies to Web Client.

However, in all cases, the selected value is written to the calc method location as an Excel date serial number.

# ComboBox calc method variable

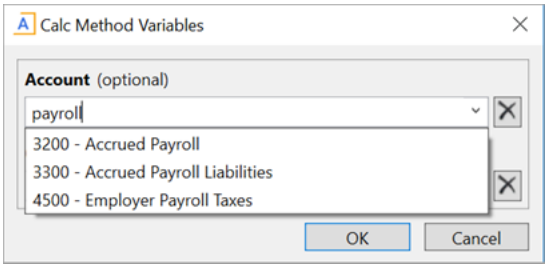
ComboBox calc method variables prompt users to select a value from a searchable drop-down list (the combo box). There are several different options to define the list of values for the combo box:

- **Using a ComboBox data source defined within the file.** ComboBox data sources support defining separate labels and values. This means that what displays to end users in the combo box can be different than what is placed in the calc method. For example, end users might select "friendly" category names (the labels) but the corresponding category code (the values) is placed in the calc method.
- **Using a table column.** You can specify a table column to display the values from that column in the drop-down list.
- **Using a picklist table.** You can specify a picklist table to display the values from that picklist in the drop-down list. This option provides special support to automatically display the picklist value to users, but write back the corresponding code as the selected value.
- **Using an Axiom Query.** You can define an Axiom query in the file, and designate this query as the source for the combo box. The values returned by the query display in the drop-down list. The query runs in the background when the user interacts with the combo box.

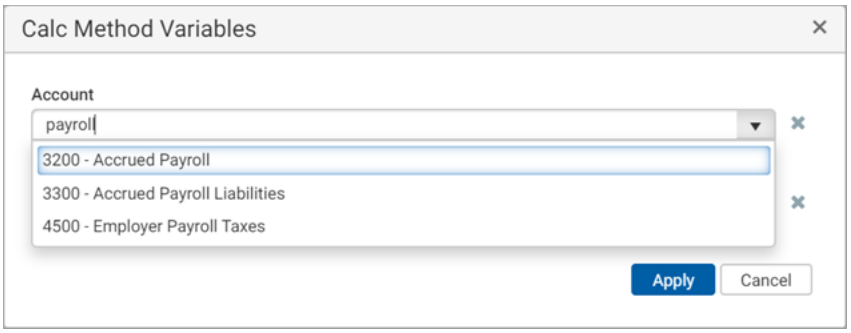
When using a ComboBox with a table column or an Axiom query, you can also use RelatedColumnValue variables. These are special variables that do not display to users. Instead, they are used to return values from related columns so that these values can be placed in the calc method. For example, if you are prompting users to select an account, then you can use a RelatedColumnValue variable to return the description of that account from the Acct.Description column. For more information, see RelatedColumnValue calc method variable.

## Variable behavior

The variable displays as a drop-down list with a searchable entry box. The user can scroll the list and select the value directly, or type into the box to find a particular value.




*Desktop Client: Example ComboBox calc method variable*




*Web Client: Example ComboBox calc method variable*

The drop-down lists may be limited to displaying some number of values (the specific display limit depends on the source of data and the client environment). For example, in some cases the drop-down list will only display the first 100 values. However, all values can be found by using the search box. The search matches on the primary value, any description columns, and any additional columns included in the display format.

The values in the drop-down list are sorted as follows:

- **ComboBox data source:** Values are presented in the same order as the data source.
- **Table column:** Values are sorted based on the display format if defined, otherwise based on the value column.
- **Picklist table**: Values are sorted based on the string value of the picklist by default. If a display format is defined, values are sorted based on the display format.
- **Axiom query:** Values are sorted according to the Axiom query **Data Sort** setting.

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a ComboBox variable. Some data source columns do not apply in this case and are not discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define ComboBox variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties (ComboBox data source)

The following additional properties apply to ComboBox variable types, when using a ComboBox data source:

| Column Tag | Description |
|---|---|
| [DataSourceName] | The name of the ComboBox data source to provide the list of values for the variable. You must define a ComboBox data source within the file in order to use this option.<br><br>The name of the data source is defined within the ComboBox tag. For example, if the tag is [ComboBox;MyName], then you would enter MyName as the data source name.<br><br>The ComboBox data source has two property columns: [Label] and [Value]. The labels define the list of values that display to users. When a user selects a label, the corresponding value is placed in the calc method location.<br><br>For more information, see Creating a ComboBox data source for the variable. |
| [PlaceholderText] | Optional. Specifies placeholder text to display within the combo box until a value is selected. If blank, then the default text "Choose a value" is used. |
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: Dept.VP='Smith'. This option is not commonly used in calc method variables.<br><br>**NOTE:** The values in the ComboBox data source are always considered to be strings and will be quoted if this option is enabled, even if the values are actually numbers. |

The following properties do not apply to ComboBox variables when using a ComboBox data source: ListChoices, ColumnName, AdditionalColumns, ColumnFilter, AllowMultiSelect, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (table column)

The following additional properties apply to ComboBox variable types, when using a table column:

| Column Tag | Description |
|---|---|
| `[ColumnName]` | The column to provide the list of values for the variable. Enter a fully-qualified Table.Column name such as `Acct.Acct`. Multi-level lookups can be used.<br><br>You can specify any column from any client table in your system. System tables such as Axiom.Aliases are not supported for use with variables and cannot be used.<br><br>When using columns with lookups (including multi-level lookups), the final lookup table is considered the primary table. For example, if you specify `GL2021.Dept`, this is the same as specifying `GL2021.Dept.Dept`, so the Dept table is the primary table. Any columns listed in filters and as additional columns must be resolvable from the primary table, or must contain a fully qualified path from the starting table (GL2021 in this example).<br><br>When using columns with lookups, the starting table impacts the list of items to be returned from the value column. For example, `GL2021.Dept` returns only the departments used in the GL2021 table, whereas `Dept.Dept` returns the full list of departments defined in the Dept table.<br><br>If the value column is a key column or a validated column, then the corresponding descriptions automatically display with the column values in the drop-down list, unless a display format is defined. |
| `[ColumnFilter]` | Optional. Specifies a filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**.<br><br>If the value column uses a lookup, then the column in the filter criteria statement must be resolvable from the primary table, or must use a fully qualified path from the starting table. |
| `[PlaceholderText]` | Optional. Specifies placeholder text to display within the combo box until a value is selected. If blank, then the default text "Choose a value for *ColumnName*" is used. |

| Column Tag | Description |
| --- | --- |
| [DisplayFormat] | Optional. Defines a display format for the items in the list, and specifies additional columns to display. By default, items in the list are displayed as:<br><br>*KeyColumn - DescriptionColumn*<br><br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br><br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br><br>This would display account items in the following format:<br><br>8000 - Facilities (Overhead)<br><br>Any columns listed should use fully qualified Table.Column syntax. If the value column uses a lookup, then any additional columns must be resolvable from the primary table, or must use a fully qualified path from the starting table.<br><br>If a display format is defined, the items in the list are sorted based on the display format instead of the value column.<br><br>**NOTE:** This is the only way to display additional columns in the list. The [AdditionalColumns] property does not apply to ComboBox variables. Additional columns included in the display format are searchable within the list. |
| [TooltipColumn] | Optional. Specifies a column that defines tooltip text for each value shown in the list. When a user hovers over a value in the list, the corresponding text from this column is shown in a tooltip.<br><br>If the value column uses a lookup, then the tooltip column must be resolvable from the primary table, or must use a fully qualified path from the starting table. |
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: Dept.VP='Smith'. This option is not commonly used in calc method variables.<br><br>**NOTE:** This option only applies if the value column is a string column. |

The following properties do not apply to ComboBox variables when using a table column: ListChoices, AllowMultiSelect, DataSourceName, DisplayFormat, Hierarchies, MinDate, MaxDate, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (Picklist table)

Although you can specify a picklist table column using the `[ColumnName]` field, the `[DataSourceName]` field provides special support for picklist tables. You can specify just the picklist table name, and the combo box will automatically display the picklist value to users for selection, while using the corresponding code as the selected value.

| Column Tag | Description |
|---|---|
| `[DataSourceName]` | Specifies the picklist table to define the list of values. Use the following syntax:<br>`Picklist: TableName`<br>For example, Picklist: Category uses the picklist table named Category.<br>When a picklist table is the data source, this is effectively the same as designating the Code column of the picklist table as the source column. However, instead of displaying the integer codes to users, by default the drop-down list displays the string values to users. Users select a string value from the list, but the selected value is the corresponding code. |
| `[ColumnFilter]` | Optional. Specifies a filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**. |
| `[PlaceholderText]` | Optional. Specifies placeholder text to display within the combo box until a value is selected. If blank, then the default text "Choose a value for *ColumnName*" is used. |

| Column Tag | Description |
|---|---|
| [DisplayFormat] | Optional. Defines a display format for the items in the list.<br><br>By default, items in the list are displayed using just the value in the Value column of the designated picklist table. The code and description are not displayed.<br><br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br><br>{Category.Value} - {Category.Description}<br><br>This would display items in the following format:<br><br>New - Use this category for new requests<br><br>The display format can use any column from the picklist table.<br><br>If a display format is defined, the items in the list are sorted based on the display format instead of the value column. |
| [TooltipColumn] | Optional. Specifies a column in the picklist table that defines tooltip text for each value shown in the list. When a user hovers over a value in the list, the corresponding text from this column is shown in a tooltip.<br><br>For example, you could specify Description to display the contents of the Description column as tooltips. |

The following properties do not apply to ComboBox variables when using a picklist table: ListChoices, AdditionalColumns, AllowMultiSelect, Hierarchies, MinDate, MaxDate, AutoQuoteString, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (Axiom query)

The following additional properties apply to ComboBox variable types, when using an Axiom query:

| Column Tag | Description |
|---|---|
| [DataSourceName] | The name of the Axiom query to provide the list of values for the variable. The sheet where the query is defined must also be specified, for example:<br><br>Sheet2!AQList<br><br>For more information on how to set up this query for use with a calc method variable, see Setting up an Axiom query for the variable. |

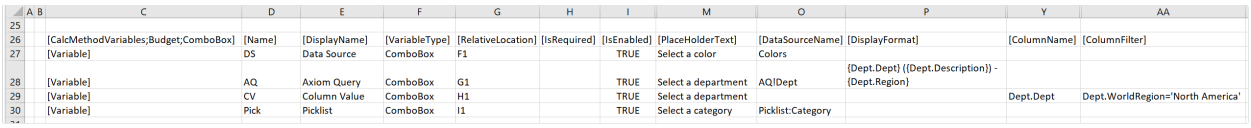
| Column Tag | Description |
| --- | --- |
| [ColumnFilter] | Optional. A filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**.<br><br>This property can be used in addition to (or instead of) the **Data Filter** on the Axiom query itself. |
| [PlaceholderText] | Optional. Specifies placeholder text to display within the combo box until a value is selected. If blank, then the default text "Choose a value for *ColumnName*" is used (where *ColumnName* is the first column in the Axiom query's field definition). |
| [DisplayFormat] | Optional. Defines a display format for the items in the list, and specifies additional columns to display. By default, items in the list are displayed as:<br><br>*KeyColumn - DescriptionColumn*<br><br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br><br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br><br>This would display account items in the following format:<br><br>8000 - Facilities (Overhead)<br><br>Any additional columns included here must also be in the field definition of the Axiom query.<br><br>**NOTE:** This is the only way to display additional columns in the combo box. The [AdditionalColumns] property does not apply to ComboBox variables. Additional columns included in the display format are searchable within the list. |
| [TooltipColumn] | Optional. Specifies a column that defines tooltip text for each value shown in the list. When a user hovers over a value in the list, the corresponding text from this column is shown in a tooltip.<br><br>The specified column must also be present in the Axiom query field definition. |

| Column Tag | Description |
|---|---|
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br><br>**NOTE:** This option only applies if the value column of the Axiom query (the first column in the field definition) is a string column. |

The following properties do not apply to ComboBox variables when using an Axiom query: ListChoices, ColumnName, AdditionalColumns, AllowMultiSelect, Hierarchies, MinDate, MaxDate, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Example data source

The following screenshot shows examples of ComboBox variables, one of each type.



| | C | D | E | F | G | H | I | M | O | P | Y | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | | | | | | | | | | | | |
| 26 | [CalcMethodVariables;Budget;ComboBox] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [PlaceHolderText] | [DataSourceName] | [DisplayFormat] | [ColumnName] | [ColumnFilter] |
| 27 | [Variable] | DS | Data Source | ComboBox | F1 | | TRUE | Select a color | Colors | | | |
| 28 | [Variable] | AQ | Axiom Query | ComboBox | G1 | | TRUE | Select a department | AQ!Dept | {Dept.Dept} {{Dept.Description}} - {Dept.Region} | | |
| 29 | [Variable] | CV | Column Value | ComboBox | H1 | | TRUE | Select a department | | | Dept.Dept | Dept.WorldRegion='North America' |
| 30 | [Variable] | Pick | Picklist | ComboBox | I1 | | TRUE | Select a category | Picklist:Category | | | |

## Creating a ComboBox data source for the variable

If you are defining calc method variables for use in a form-enabled file, then you can use the data source wizard to add the data source. Right-click in the cell where you want to start the data source, then select **Create Axiom Form Data Source > Combo Box**.

If you are defining variables for use in a spreadsheet Axiom file, then you must manually create the data source.

The tags for the Combo Box data source are as follows:

### Primary tag

**[ComboBox;*DataSourceName*]**

The DataSourceName identifies this data source. Data source names must be unique within a file and must start with a letter. Names can only contain letters, numbers, and underscores. Names are validated when the file is saved; an invalid name will prevent the save.

The placement of this primary tag defines the control column and the control row for the data source.

- All column tags must be placed in this row, to the right of the tag.
- All row tags must be placed in this column, below the tag.

### Row tags

**[ComboItem]**

Each row flagged with this tag defines an item to display in the combo box.

### Column tags

**[Label]**

The display name for each item in the list. Labels should be unique. If multiple rows have the same label, then the first value with that label is used.

**[Value]**

The corresponding value for each label. This can be the same value as the label, or a different value.

For example, in a list of colors, both the label and the value can be the text Blue. Or, the label text can be Blue while the value is a numeric color code. Separating the label from the value allows you to display "friendly" text to end users but use any value as the selected value.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

The following example shows how a ComboBox data source might look in a file:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | [ComboBox;Regions] | [Label] | [Value] |
| 3 | | [ComboItem] | Consolidated | All |
| 4 | | [ComboItem] | West | West |
| 5 | | [ComboItem] | North | North |
| 6 | | [ComboItem] | South | South |
| 7 | | [ComboItem] | East | East |

In this example, if the user selects the label "Consolidated" from the combo box, the value "All" will be placed in the calc method.

### Setting up an Axiom query for the variable

When the user interacts with the combo box to select an item, the specified Axiom query is run *in memory only* (meaning, no values are populated within the sheet where the query is configured). The results of the query are used to populate the list.

The Axiom query should be set up as follows:

- The first column in the field definition is the value column for the variable—meaning the values to be placed in the relative location of the calc method. If the value column is a key column, then the second column in the field definition is the description column for the key values.
- The field definition of the query must also contain any additional columns used in the `[DisplayFormat]` property, as well as any columns used by associated RelatedColumnValue variables. These columns must be placed after the key and description columns. All columns in the field definition must be contiguous (no blank cells in between).
- It is recommended to use fully qualified Table.Column names in the field definition for the Axiom query. If you define a display format for the variable or use a dependent RelatedColumnValue variable, both of those features require the columns to be fully qualified, and they must match the field definition entries exactly.
- The Axiom query data filter can be used to filter the list of values. If desired, you can also (or alternatively) use the `[ColumnFilter]` property of the RefreshVariables data source.
- Although the query itself must be active, all refresh behavior options for the Axiom query should be set to **Off** (such as **Refresh on file open**, **Refresh on manual refresh**, etc.), unless you also want the query to run at those times for reasons other than the combo box.
- No Axiom query settings that impact the display in the sheet will apply to the combo box. This includes spreadsheet sorting (use data sort instead), in-sheet calc method formatting or formulas, and data range filters. The only Axiom query settings read from the sheet are the field definition entries. One way to think of it is that the values for the drop-down list are basically the same values that you see when using the **Preview Axiom Query Data** button on the Sheet Assistant.

- System tables such as Axiom.Columns are not supported for use in calc method variables, and cannot be used in the Axiom query.

## Grid calc method variable

Grid calc method variables prompt users to select a value from a designated table column. For example, you may want the user to select an account from the Acct.Acct column. The list of values is displayed in a dialog, using a searchable grid.

There are two ways that you can specify the column to use for the Grid variable:

- Specify the table column directly, from any table.
- Specify a Picklist table name. This automatically uses the Code column as the value column, but also automatically displays the Value and Description columns.

Both ComboBox variables and Grid variables can be used to select a value from a table column. The primary difference between the two variable types is the user interface for selecting the value—using either a drop-down list or a grid dialog. Additionally, only Grid variables allow selection of multiple values.

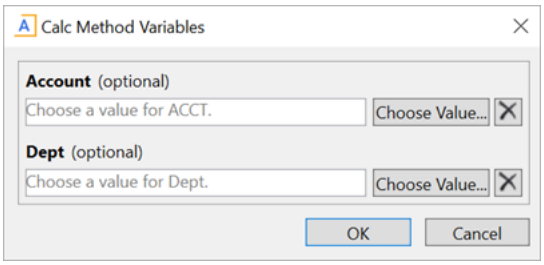
When using Grid variables, you can also use RelatedColumnValue variables. These are special variables that do not display to users. Instead, they are used to return values from related columns. For example, if you are prompting users to select an account, then you can use a RelatedColumnValue variable to return the description of that account from the Acct.Description column. For more information, see RelatedColumnValue calc method variable.

**NOTE:** Grid variables can be used when inserting calc methods in Axiom forms, but only when multi-select is enabled. If multi-select is not enabled, then the Grid variable will behave like a ComboBox variable, so it is recommended to use a ComboBox variable instead.

### Variable behavior

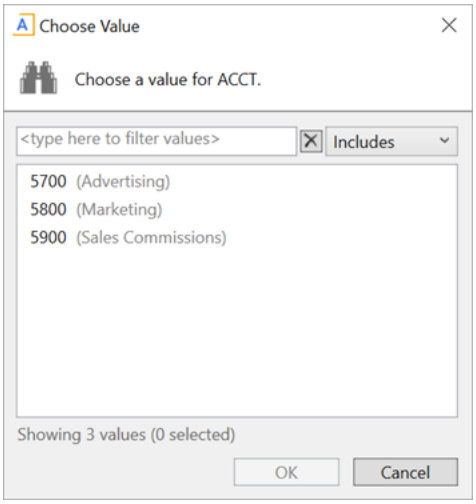
In the Desktop Client, the variable displays as a text box with a **Choose Value** button next to it. The user can do one of the following to specify a value:

- Type a value directly into the text box. The entry must match a value in the designated table column for the variable. If the user enters an invalid value, then a validation message displays at the bottom of the dialog. The **OK** button is disabled until the user enters a valid value.
- Click the button to open the **Choose Value** dialog, to select a value from the designated table column for the variable. This dialog has a search box so that the user can type to find values in lengthy lists. If multi-select is enabled for the variable, then the Choose Value dialog has check boxes to enable multiple selections. Once the user has selected a value, the value displays in the text box.

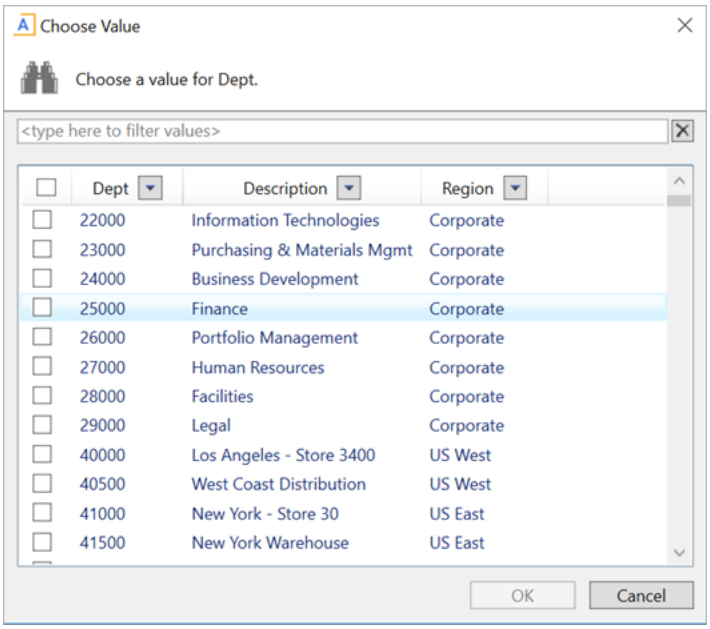



*Desktop Client: Example grid variables*

The Choose Value dialog uses either the "simple view" or the "full grid view" depending on the data to be displayed in the dialog. If only one column is being shown, or a key column plus description only, then the simple view is used. However, if the target column is a key column and any additional non-description columns are included, then the full grid view is used.




*Example Choose Value dialog using "simple view"*

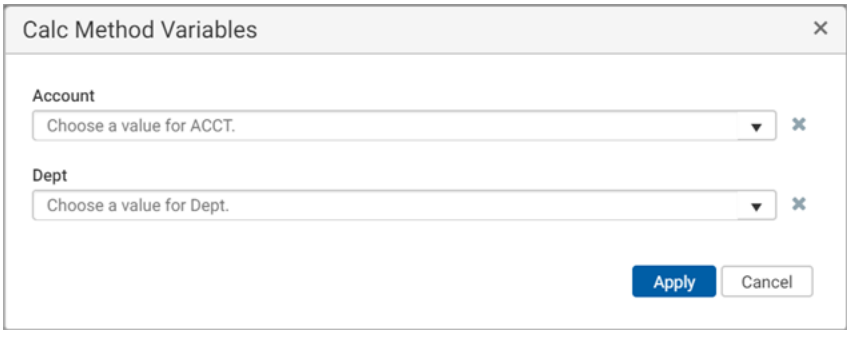



*Example Choose Value dialog using "full grid view" and with multi-select enabled*

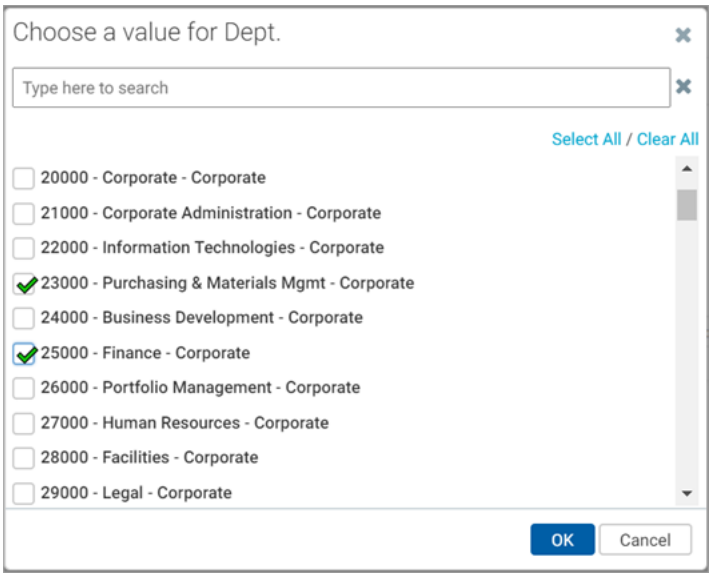
The list of values in the Choose Value dialog is sorted based on the designated table column for the variable. In some cases the values may not be displayed as expected:

- If the column is from a document reference table, the values will most likely not be displayed in the order they are stored in the source file. When the values are saved to the database from the source file, they are sorted based on the key column of the table (column A of the sheet).
- If the column is a string column that contains numeric values (values that are all numbers and values that start with numbers), Axiom will attempt to sort these values in numeric order. Values that start with letters will then be sorted in alphabetical order.

When used with Axiom forms and multi-select is enabled for the variable, the variable initially looks like a ComboBox variable.




However, when the user clicks the down arrow to select a value, a dialog opens instead, allowing the user to select multiple values using check boxes. The behavior of this dialog is the same as when using multi-select with a form combo box (Select tag).




The "full grid" view is not used with Axiom forms, even if multiple additional columns are specified. The additional column values are concatenated with hyphens. This means that the values cannot be sorted by column, but users can still use the search box to find particular values.

If multi-select is not enabled for the variable, then the variable behaves the same way as a ComboBox variable when used in Axiom forms. Therefore, it is strongly recommended to use a ComboBox variable with Axiom forms instead, if you do not need multi-select.

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a Grid variable. Some data source columns do not apply in this case and are not discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define Grid variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties (table column)

The following additional properties apply to Grid variable types, when specifying a table column directly.

| Column Tag | Description |
| --- | --- |
| `[ColumnName]` | The column to provide the list of values for the variable. Enter a fully-qualified Table.Column name such as `Acct.Acct`. Multi-level lookups can be used.<br><br>You can specify any column from any client table in your system. System tables such as Axiom.Aliases are not supported for use with variables and cannot be used.<br><br>When using columns with lookups (including multi-level lookups), the final lookup table is considered the primary table. For example, if you specify `GL2021.Dept`, this is the same as specifying `GL2021.Dept.Dept`, so the Dept table is the primary table. Any columns listed in filters and as additional columns must be resolvable from the primary table, or must contain a fully qualified path from the starting table (GL2021 in this example).<br><br>When using columns with lookups, the starting table impacts the list of items to be returned from the value column. For example, `GL2021.Dept` returns only the departments used in the GL2021 table, whereas `Dept.Dept` returns the full list of departments defined in the Dept table.<br><br>If the value column is a key column or a validated column, then the corresponding descriptions automatically display with the column values in the grid, unless additional columns are defined. |

| Column Tag | Description |
| --- | --- |
| [AdditionalColumns] | Optional. One or more additional columns to display in the selection dialog along with the value column. Separate multiple column names with commas or semicolons.<br><br>Any columns listed should use fully qualified Table.Column syntax. If the value column uses a lookup, then any additional columns must be resolvable from the primary table, or must use a fully qualified path from the starting table. Column-only syntax is only allowed for columns directly on the primary table.<br><br>**NOTES:**<br>• If [AdditionalColumns] is left blank, then any description columns associated with the value column are automatically included in the grid. However, if you do specify any additional columns then you must also include the desired description columns.<br>• When used in the Web Client, additional column values are concatenated to the key value using hyphens. They do not display as separate columns. For example, if the key is Dept and additional columns are Description and Country, the value will display like 24000 - Finance - United States. If you do not like this display, then in the Web Client only, you can use the [DisplayFormat] parameter to impact the display of the list in the multi-select dialog. |
| [ColumnFilter] | Optional. A filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**.<br><br>If the value column uses a lookup, then the column in the filter criteria statement must be resolvable from the primary table, or must use a fully qualified path from the starting table. |
| [PlaceholderText] | Optional. Defines placeholder text to display within the variable box until a value is selected. This text also displays at the top of the **Choose Value** dialog, and as a tooltip for the **Choose Value** button. If blank, then the default text "Choose a value for *ColumnName*" is used. |

| Column Tag | Description |
| --- | --- |
| [AllowMultiSelect] | Optional. Specifies whether multiple values can be selected from the designated column (True/False).<br>• If True, then the list of values will display with check boxes and the user can select more than one item. The values will be placed in the calc method location as a comma-separated list. If the column is a string column, the values will automatically be wrapped in single quotation marks.<br>• If blank or False, then only one value can be selected from the column. |
| [DisplayFormat] | Optional. Defines a display format for the items in the list. This only applies to Grid variables when used in the Web Client, to control the display of items in the multi-select dialog. The display format is ignored in the Desktop Client.<br>By default, items in the list are displayed as:<br>*KeyColumn - DescriptionColumn*<br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br>This would display account items in the following format:<br>8000 - Facilities (Overhead)<br>Any columns listed should use fully qualified Table.Column syntax. If the value column uses a lookup, then any additional columns must be resolvable from the primary table, or must use a fully qualified path from the starting table.<br>If a display format is defined, the items in the list are sorted based on the display format instead of the value column. |
| [TooltipColumn] | Optional. Specifies a column that defines tooltip text for each value shown in the list. When a user hovers over a value in the list, the corresponding text from this column is shown in a tooltip.<br>If the value column uses a lookup, then the tooltip column must be resolvable from the primary table, or must use a fully qualified path from the starting table. |

| Column Tag | Description |
|---|---|
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br><br>**NOTE:** This option only applies if the value column is a string column, and only if multi-select is not enabled for the variable. If multi-select is enabled, then string values are always quoted. |

The following properties do not apply to Grid variables when using a table column: DataSourceName, ListChoices, Hierarchies, MinDate, MaxDate, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (Picklist table)

The following additional properties apply to Grid variable types, when specifying a Picklist table.

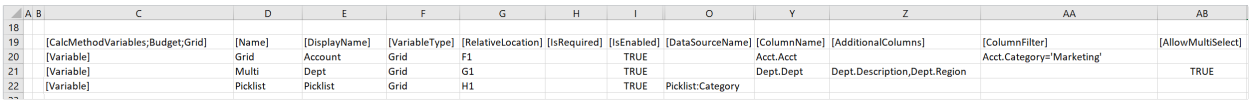
| Column Tag | Description |
|---|---|
| [DataSourceName] | Specifies the picklist table to define the list of values. Use the following syntax:<br><br>`Picklist: TableName`<br><br>For example, `Picklist: Category` uses the picklist table named Category.<br><br>When a picklist table is the data source, this is effectively the same as designating the Code column of the picklist table as the source column. The additional picklist columns of Value and Description are also displayed in the grid. |
| [ColumnFilter] | Optional. A filter criteria statement to limit the list of codes displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**. |
| [PlaceholderText] | Optional. Defines placeholder text to display within the variable box until a code is selected. This text also displays at the top of the **Choose Value** dialog, and as a tooltip for the **Choose Value** button. If blank, then the default text "Choose a value for *ColumnName*" is used. |

| Column Tag | Description |
|---|---|
| [AllowMultiSelect] | Optional. Specifies whether multiple codes can be selected from the picklist (True/False).<br>• If True, then the list of codes will display with check boxes and the user can select more than one item. The codes will be placed in the calc method location as a comma-separated list.<br>• If blank or False, then only one code can be selected from the column. |
| [DisplayFormat] | Optional. Defines a display format for the items in the list. This only applies to Grid variables when used in the Web Client, to control the display of items in the multi-select dialog. The display format is ignored in the Desktop Client.<br>By default, items in the list are displayed as:<br>*KeyColumn - DescriptionColumn*<br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br>This would display account items in the following format:<br>8000 - Facilities (Overhead)<br>If a display format is defined, the items in the list are sorted based on the display format instead of the value column.<br>When defining a display format for use in the Web Client, you can also optionally use the TooltipColumn property to display tooltip text for each item. This is not applicable in the Desktop Client, because all picklist columns display in the grid in the Desktop Client. |
| [TooltipColumn] | Optional. Specifies a column in the picklist table that defines tooltip text for each value shown in the list. When a user hovers over a value in the list, the corresponding text from this column is shown in a tooltip.<br>Although you can specify a tooltip column when using a picklist table data source in the Desktop Client, there is typically no reason to do so, because the grid already shows all columns in the picklist table. |

The following properties do not apply to Grid variables when using a picklist table: ListChoices, ColumnName, AdditionalColumns, Hierarchies, MinDate, MaxDate, AutoQuoteString, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Example data source

The following screenshot shows examples of Grid variables. The first variable uses a filter and does not allow multi-selection. The second variable allows multi-selection. The third variable uses a picklist table.



| | C | D | E | F | G | H | I | O | Y | Z | AA | AB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | | | | | | | | | | | | |
| 19 | [CalcMethodVariables;Budget;Grid] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [DataSourceName] | [ColumnName] | [AdditionalColumns] | [ColumnFilter] | [AllowMultiSelect] |
| 20 | [Variable] | Grid | Account | Grid | F1 | | TRUE | | Acct.Acct | | Acct.Category='Marketing' | |
| 21 | [Variable] | Multi | Dept | Grid | G1 | | TRUE | | Dept.Dept | Dept.Description,Dept.Region | | TRUE |
| 22 | [Variable] | Picklist | Picklist | Grid | H1 | | TRUE | Picklist:Category | | | | |

## GUID calc method variable

GUID variables can be used to generate a globally unique identifier (GUID) when a calc method is inserted. This is a special variable type that does not display to the user in the Calc Method Variables dialog—it is only used to generate the ID and place it into the inserted calc method. For example, when inserting calc methods for planning new hires, you may want to generate a unique ID for each new hire.

### Variable behavior

GUID variables do not display in the Calc Method Variables dialog. If other variables are defined for the calc method, then the variables dialog is shown to the user but the GUID variable is omitted. If the GUID variable is the only variable, then the variables dialog is not shown to the user.

When the calc method is inserted, the designated relative location for the GUID variable is populated with an automatically generated GUID string.

For example, imagine that you have the following calc method configuration. This example uses the CalcMethodVariables data source, but the general configuration and behavior is the same when using legacy variables.

| | A | B | C | D | F | G | I |
|---|---|---|---|---|---|---|---|
| 3 | | | | | | | |
| 4 | | | [CalcMethodVariables;Payroll;New Employee] | [Name] | [VariableType] | [RelativeLocation] | [IsEnabled] |
| 5 | | | [Variable] | ID | GUID | E1 | TRUE |

When the user inserts this calc method, no variables dialog displays to the user. Instead the calc method is directly inserted, and a GUID is generated and placed in column E, in the first row of the calc method.

| EmpID | Name | Type | Rate | Increase | Override | TOTAL | Desc | BUDGET Jan | BUDGET Feb |
|---|---|---|---|---|---|---|---|---|---|
| 75341078-24fb-40d9-b0c3-e4f35956693e | [Employee Name] | Hourly | | | | - | Hrs / Week | 40.00 | 40.00 |
| | Developer | | | | | | Rate | - | - |
| | TCHNCL | Hire Month | Jan-17 | | | | Wages | - | - |
| | | Starting Wage | 0.00 | | | | Bonus | - | - |

The ID is alphanumeric with dashed sections, such as: a635e1cb-4001-46bd-a427-a704e29b58b8. This format cannot be customized, though you can manipulate the format as needed in the spreadsheet after insertion. If you are saving the generated ID to the database, the destination column must be a string column with a minimum width of 36.

### ▶ Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a GUID variable. Some data source columns do not apply in this case and are not discussed here. If these inapplicable columns are present in the data source, they should be left blank on rows that define GUID variable types. For more information on the GUID data source in general, see Defining calc method variables using a CalcMethodVariables data source.

#### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

#### Variable-specific properties

GUID variables do not have any variable-specific properties. None of the other data source properties apply to the variable.

## RadioButton calc method variable

RadioButton calc method variables prompt users to select a value from two or more radio buttons. There are three ways to define the list of values to be displayed as radio buttons. These are the same three options supported by ComboBox variables:

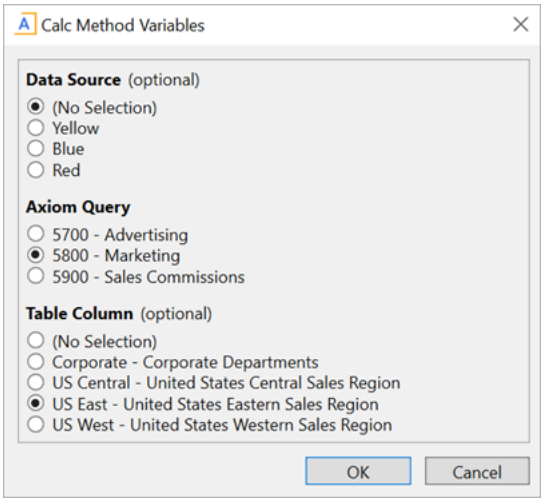
- ComboBox data source
- Table column
- Axiom query

The RadioButton variable should only be used for small lists of values. Large lists are difficult for users to read in radio button format, and take up too much space in the Calc Method Variables dialog. If the list is too large to display effectively in radio button format, then you should use a different variable type, such as StringList, Grid, or ComboBox.

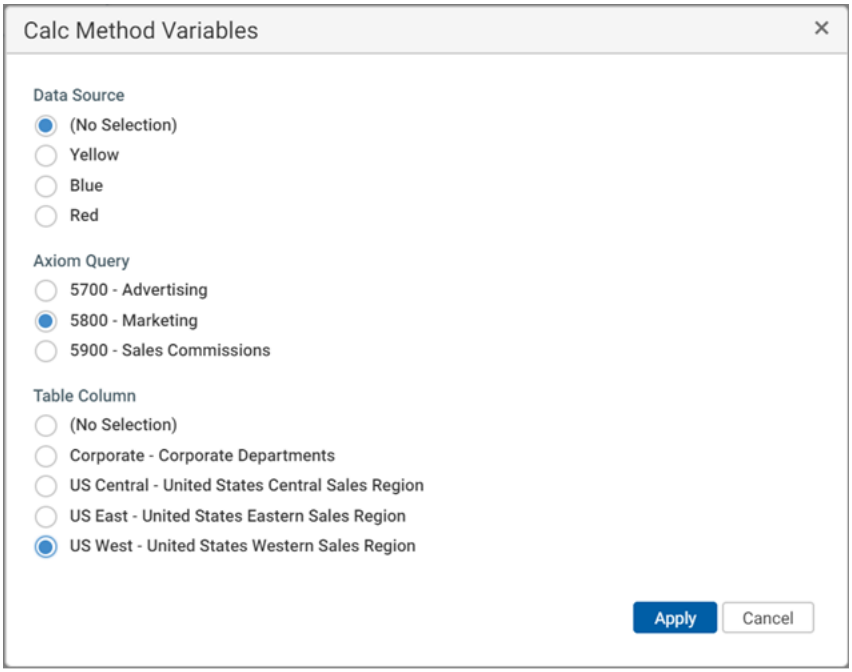
Unlike ComboBox variables, RadioButton variables do not support use of associated RelatedColumnValue variables. This is because the kind of data displayed as radio buttons typically does not have related column values that need to be brought into the spreadsheet. Although ComboBox variables and RadioButton variables use the same data options, the kind of data they use is very different.

## Variable behavior

The variable displays as a vertical list of radio buttons. The user can select one of the radio buttons to use that value. The following example shows three different radio button variables.




*Desktop Client: Example RadioButton variables*




*Web Client: Example RadioButton variables*

If the variable is not required, the user can select **(No Selection)**. This means that the selected value for the variable will be blank. The (No Selection) option is automatically added to the top of optional variables; you do not need to manually add this value to your list of values.

If the variable is required, then the user must select one of the radio buttons. The (No Selection) option is not available.

The radio button values are presented in the following order, depending on the source data:

- **ComboBox data source:** Values are presented in the same order as the data source.
- **Table column:** Values are sorted based on the display format if defined, otherwise based on the value column.
- **Axiom query:** Values are sorted according to the Axiom query **Data Sort** setting.

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a RadioButton variable. Some data source columns do not apply in this case and are not discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define RadioButton variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

The remaining columns depend on whether you are using a ComboBox data source, a table column, or an Axiom query. Because RadioButton variables use small lists of values, the most common way to define the list of values is to use a ComboBox data source. If you use a table column or an Axiom query to define the list of values, you must ensure that the returned list of values is small.

### Variable-specific properties (ComboBox data source)

The following additional properties apply to RadioButton variable types, when using a ComboBox data source:

| Column Tag | Description |
|---|---|
| [DataSourceName] | The name of the ComboBox data source to provide the list of values for the variable. You must define a ComboBox data source within the file in order to use this option.<br><br>The name of the data source is defined within the ComboBox tag. For example, if the tag is [ComboBox;MyName], then you would enter MyName as the data source name.<br><br>The ComboBox data source has two property columns: [Label] and [Value]. The labels define the list of values that display to users. When a user selects a label, the corresponding value is placed in the calc method location.<br><br>For more information, see Creating a ComboBox data source for the variable. |
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: Dept.VP='Smith'. This option is not commonly used in calc method variables.<br><br>**NOTE:** The values in the ComboBox data source are always considered to be strings and will be quoted if this option is enabled. |

The following properties do not apply to RadioButton variables when using a ComboBox data source: PlaceHolderText, ListChoices, ColumnName, AdditionalColumns, ColumnFilter, AllowMultiSelect, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (table column)

The following additional properties apply to RadioButton variable types, when using a table column:

| Column Tag | Description |
| --- | --- |
| [ColumnName] | The column to provide the list of values for the variable. Enter a fully-qualified Table.Column name such as Acct.Acct. Multi-level lookups can be used.<br><br>You can specify any column from any client table in your system. System tables such as Axiom.Aliases are not supported for use with refresh variables and cannot be used.<br><br>When using columns with lookups (including multi-level lookups), the final lookup table is considered the primary table. For example, if you specify GL2021.Dept, this is the same as specifying GL2021.Dept.Dept, so the Dept table is the primary table. Any columns listed in filters and as additional columns must be resolvable from the primary table, or must contain a fully qualified path from the starting table (GL2021 in this example).<br><br>When using columns with lookups, the starting table impacts the list of items to be returned from the value column. For example, GL2021.Dept returns only the departments used in the GL2021 table, whereas Dept.Dept returns the full list of departments defined in the Dept table.<br><br>If the value column is a key column or a validated column, then the corresponding descriptions automatically display with the column values in the drop-down list, unless a display format is defined. |
| [ColumnFilter] | Optional. A filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**.<br><br>If the value column uses a lookup, then the column in the filter criteria statement must be resolvable from the primary table, or must use a fully qualified path from the starting table. |

| Column Tag | Description |
|---|---|
| [DisplayFormat] | Optional. Defines a display format for the items in the list, and specifies additional columns to display. By default, items in the list are displayed as:<br><br>*KeyColumn - DescriptionColumn*<br><br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br><br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br><br>This would display account items in the following format:<br><br>8000 - Facilities (Overhead)<br><br>Any columns listed should use fully qualified Table.Column syntax. If the value column uses a lookup, then any additional columns must be resolvable from the primary table, or must use a fully qualified path from the starting table.<br><br>If a display format is defined, the items in the list are sorted based on the display format instead of the value column. |
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br><br>**NOTE:** This option only applies if the value column is a string column. |

The following properties do not apply to ComboBox variables when using a table column: PlaceHolderText, ListChoices, AdditionalColumns, ColumnFilter, AllowMultiSelect, DataSourceName, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, AutoQuoteString, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Variable-specific properties (Axiom query)

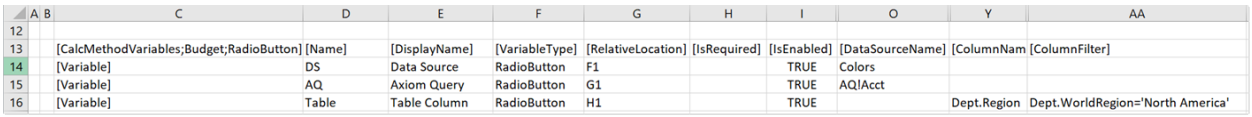
The following additional properties apply to RadioButton variable types, when using an Axiom query:

| Column Tag | Description |
|---|---|
| [DataSourceName] | The name of the Axiom query to provide the list of values for the variable. The sheet where the query is defined must also be specified, for example:<br>`Sheet2!AQList`<br>For more information on how to set up this query for use with a calc method variable, see Setting up an Axiom query for the variable. |
| [ColumnFilter] | Optional. A filter criteria statement to limit the list of values displayed to the user. You can type in the filter statement manually, or right-click the cell and use **Axiom Wizards > Filter Wizard**.<br>This property can be used in addition to (or instead of) the **Data Filter** on the Axiom query itself. |
| [DisplayFormat] | Optional. Defines a display format for the items in the list, and specifies additional columns to display. By default, items in the list are displayed as:<br>*KeyColumn - DescriptionColumn*<br>If you want to specify a different format and/or use additional columns, then you can indicate the display format here. Use fully qualified Table.Column syntax and place column references in curly brackets. For example, you could indicate something like:<br>{Acct.Acct} - {Acct.Description} ({Acct.Category})<br>This would display account items in the following format:<br>8000 - Facilities (Overhead)<br>Any additional columns included here must also be in the field definition of the Axiom query. |
| [AutoQuoteString] | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br>**NOTE:** This option only applies if the value column of the Axiom query (the first column in the field definition) is a string column. |

The following properties do not apply to Radio Button variables when using an Axiom query: PlaceHolderText, ListChoices, ColumnName, AdditionalColumns, AllowMultiSelect, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Example data source

The following screenshot shows examples of RadioButton variables, one of each type (data source, Axiom query, and column value).



| | A | B | C | D | E | F | G | H | I | O | Y | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | | | | | | | | | | | | |
| 13 | | | [CalcMethodVariables;Budget;RadioButton] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [DataSourceName] | [ColumnNam | [ColumnFilter] |
| 14 | | | [Variable] | DS | Data Source | RadioButton | F1 | | TRUE | Colors | | |
| 15 | | | [Variable] | AQ | Axiom Query | RadioButton | G1 | | TRUE | AQlAcct | | |
| 16 | | | [Variable] | Table | Table Column | RadioButton | H1 | | TRUE | | Dept.Region | Dept.WorldRegion='North America' |

## Creating a ComboBox data source for the variable

If you are defining calc method variables for use in a form-enabled file, then you can use the data source wizard to add the ComboBox data source. Right-click in the cell where you want to start the data source, then select **Create Axiom Form Data Source > Combo Box**.

If you are defining calc method variables for use in a spreadsheet Axiom file, then you must manually create the data source.

The tags for the Combo Box data source are as follows:

### Primary tag

**`[ComboBox;DataSourceName]`**

The DataSourceName identifies this data source. Data source names must be unique within a file and must start with a letter. Names can only contain letters, numbers, and underscores. Names are validated when the file is saved; an invalid name will prevent the save.

The placement of this primary tag defines the control column and the control row for the data source.

- All column tags must be placed in this row, to the right of the tag.
- All row tags must be placed in this column, below the tag.

### Row tags

**`[ComboItem]`**

Each row flagged with this tag defines an item to display in the combo box.

### Column tags

**`[Label]`**

The display name for each item in the list. Labels should be unique. If multiple rows have the same label, then the first value with that label is used.

**[Value]**

The corresponding value for each label. This can be the same value as the label, or a different value.

For example, in a list of colors, both the label and the value can be the text Blue. Or, the label text can be Blue while the value is a numeric color code. Separating the label from the value allows you to display "friendly" text to end users but use any value as the selected value.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

The following example shows how a ComboBox data source might look in a file:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | [ComboBox;Regions] | [Label] | [Value] |
| 3 | | [ComboItem] | Consolidated | All |
| 4 | | [ComboItem] | West | West |
| 5 | | [ComboItem] | North | North |
| 6 | | [ComboItem] | South | South |
| 7 | | [ComboItem] | East | East |

In this example, if the user selects the radio button labeled "Consolidated", then the value "All" will be placed in the `[SelectedValue]` column.

## Setting up an Axiom query for the variable

When the RadioButton variable is rendered in the Calc Method Variables dialog, the specified Axiom query is run *in memory only* (meaning, no values are populated within the sheet where the query is configured). The results of the query are used to generate the list of radio buttons.

The Axiom query should be set up as follows:

- The first column in the field definition is the value column for the variable—meaning the values to be placed in the calc method location. If the value column is a key column, then the second column in the field definition is the description column for the key values.
- The field definition of the query must also contain any additional columns used in the `[DisplayFormat]` property. These columns must be placed after the key and description columns. All columns in the field definition must be contiguous (no blank cells in between).

- It is recommended to use fully qualified Table.Column names in the field definition for the Axiom query. If you define a display format for the variable, the display format must use fully qualified columns, and they must match the field definition entries exactly.
- The Axiom query data filter can be used to filter the list of values. If desired, you can also (or alternatively) use the `[ColumnFilter]` property of the CalcMethodVariables data source.
- All refresh behavior options for the Axiom query should be set to **Off** (such as **Refresh on file open**, **Refresh on manual refresh**, etc.), unless you also want the query to run at those times for reasons other than the refresh variable.
- No Axiom query settings that impact the display in the sheet will apply to the variable. This includes spreadsheet sorting (use data sort instead), in-sheet calc method formatting or formulas, and data range filters. The only Axiom query settings read from the sheet are the field definition entries. One way to think of it is that the values for the drop-down list are basically the same values that you see when using the **Preview Axiom Query Data** button on the Sheet Assistant.

## RelatedColumnValue calc method variable

RelatedColumnValue calc method variables can be used to bring in additional column values based on the user's selection for a parent variable. This is a special variable type that does not display to the user—it is only used to return values and then place them in designated calc method locations.

For example, imagine that you have a Grid variable for the user to select an account from Acct.Acct. You want to display both the account number and the account description in the inserted calc method. You can define a RelatedColumnValue variable that returns the associated value from Acct.Description for the selected account. Both values will then be placed in the calc method.

The RelatedColumnValue variable provides an alternative to using GetData functions to return these associated values. Avoiding use of unnecessary GetData functions can improve file performance.

The following variable types can be parent variables for a RelatedColumnValue variable:

- Grid variables
- ComboBox variables using a table column
- ComboBox variables using an Axiom query

The parent variable must use a key column of a reference table as its value column in order to bring in related column values.

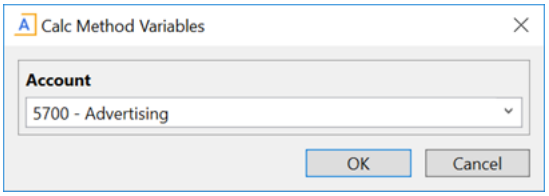
### Variable behavior

RelatedColumnValue variables do not display to users. Instead, they are automatically populated based on the user's selection for a parent variable. Once the user has completed the selections for the other calc method variables and the calc method is inserted, the relative location for the RelatedColumnValue variable is also populated with the appropriate value from the specified column for the variable.

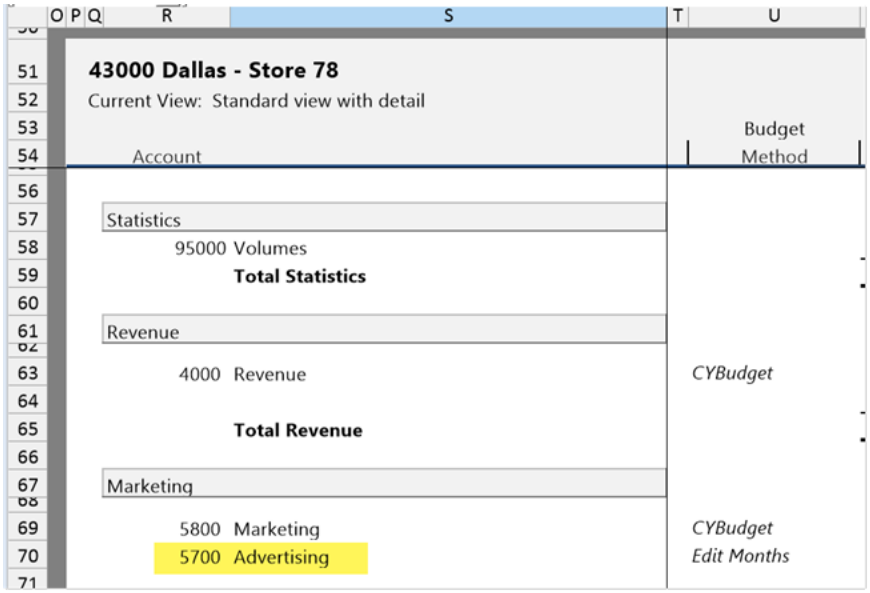
For example, imagine that you are prompting users to select an account to place in the calc method, and you also want to place the account description in the calc method. To do this, you first set up the parent variable to display a list of accounts. Then, you create a RelatedColumnValue variable to return the account description, and configure it as dependent on the parent Account variable.

| | A | B | C | D | E | F | G | H | I | K | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 27 | | | | | | | | | | | |
| 28 | | | [CalcMethodVariables;Budget;Edit Months] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [DependsOn] | [ColumnName] |
| 29 | | | [Variable] | Acct | Account | ComboBox | R1 | TRUE | TRUE | | acct.acct |
| 30 | | | [Variable] | Desc | | RelatedColumnValue | S1 | | TRUE | Acct | acct.description |

When the user inserts this calc method and the Calc Method Variable dialog displays, only the Account variable is visible.




When the calc method is inserted, Axiom finds the specified column for the RelatedColumnValue variable—in this example, Acct.Description—and returns the value for the selected account. Both values are then placed into the specified relative locations within the calc method.




## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a RelatedColumnValue variable. Some data source columns do not apply in this case and are not

discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define RelatedColumnValue variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties

The following additional properties apply to RelatedColumnValue variable types:

| Column Tag | Description |
|---|---|
| `[ColumnName]` | Specifies the related column from which to look up a value, based on the selected value for the parent variable. Enter a fully-qualified Table.Column name such as `Acct.Description`.<br><br>The column for the parent variable must be a key column of a reference table or a validated column.<br><br>• If the parent variable column is the key column of a reference table, then the related column can be any column from the same table as the parent variable, or from a table that the parent variable has a lookup relationship with. For example, if the parent variable column is `Dept.Dept`, then the related column can be any other column in that table such as `Dept.Description` or `Dept.Region`. It can also be `Dept.Region.RegionType`, assuming that `Dept.Region` looks up to `Region.Region`.<br>• If the parent variable column is a validated column, then the related column can be any column from the lookup table. For example, if the parent variable column is `Dept.Region`, then the related column could be `Dept.Region.RegionType`.<br><br>If the parent variable is a ComboBox variable that uses an Axiom query, then the related column must be present in the field definition of that query. |

| Column Tag | Description |
|---|---|
| `[AutoQuoteString]` | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br><br>**NOTE:** This option only applies if the related column is a string column. |

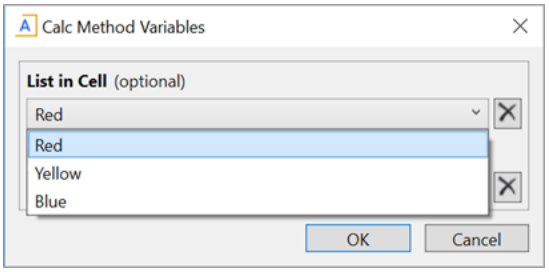
The following properties do not apply to RelatedColumnValue variables: PlaceHolderText, ListChoices, AdditionalColumns, ColumnFilter, AllowMultiSelect, DataSourceName, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

## StringList calc method variable

StringList calc method variables prompt users to make a selection from a predefined list. This list is manually defined within the current file—either in the CalcMethodVariables data source itself, or in a range of cells that are then referenced in the data source.

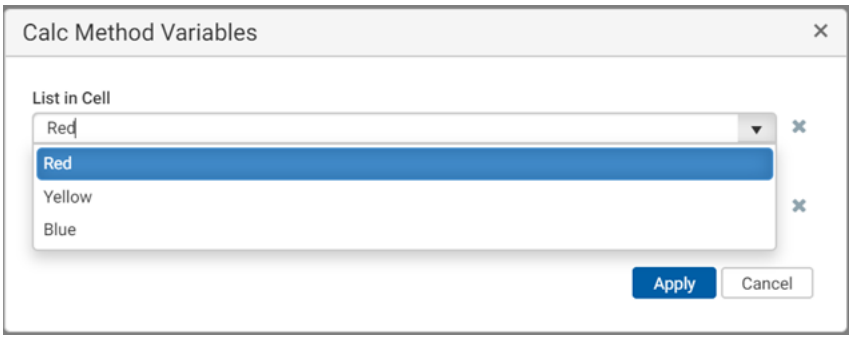
### Variable behavior

In the Desktop Client, the variable displays as a drop-down list. The user must use the drop-down to select a value.




*Desktop Client: Example StringList variable*

In the Web Client, the variable displays as a drop-down list with a searchable entry box. The user can scroll the list and select the value directly, or type into the box to find a particular value.




*Web Client: Example StringList variable*

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a StringList variable. Some data source columns do not apply in this case and are not discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define StringList variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties

The following additional properties apply to StringList variable types:

| Column Tag | Description |
|---|---|
| `[ListChoices]` | Defines the list of values for the variable. Enter one of the following:<br>• A comma-separated list of values to display in the list. For example: `Yellow, Red, Blue`. (Alternatively, semicolons can be used as the delimiter.)<br>**NOTE:** This list can contain cell references in brackets. The value for the list will be read from the designated cell. For example, you can enter: `Yellow, [List!B5], Blue`. If List!B5 contains the value Red, then the list will be: Yellow, Red, Blue. If List!B5 contains a cell range, then the values within that range will be added to the list.<br>• A cell range, in brackets. For example: `[List!B10:B20]`. The list of values will be read from this range.<br>When using a cell range, the range must be one dimensional—meaning either one column wide and several rows tall, or several columns wide and one row tall. If the range is a block, it is ignored. |
| `[PlaceholderText]` | Optional. Specifies placeholder text to display within the variable box until a value is selected. If blank, then the default text "Select..." is used.<br>For StringList variables, this property is only supported for use in the Web Client. In the Desktop Client, the variable box is blank if no value has been selected. |
| `[AutoQuoteString]` | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br>**NOTE:** All values in the list are considered strings and will be quoted if this option is enabled, even if the list values are actually numbers. |

The following properties do not apply to StringList variables: ColumnName, AdditionalColumns, ColumnFilter, AllowMultiSelect, DataSourceName, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

### Example data source

The following screenshot shows a couple of StringList variable examples. The first example uses a comma-separated list, while the second example reads the list from a cell range.

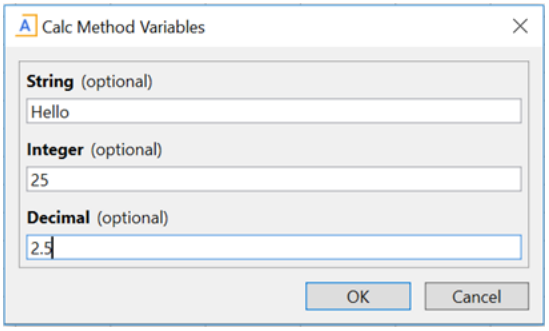
| | A | B | C | D | E | F | G | I | N |
|---|---|---|---|---|---|---|---|---|---|
| 7 | | | | | | | | | |
| 8 | | | [CalcMethodVariables;Budget;ListTest] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsEnabled] | [ListChoices] |
| 9 | | | [Variable] | Cell | List in Cell | StringList | F1 | TRUE | Red, Yellow, Blue |
| 10 | | | [Variable] | Bracket | List in Bracket | StringList | G1 | TRUE | [List!A1:A3] |

## String, Integer, and Decimal calc method variables

String, Integer, or Decimal calc method variables prompt users to make a "free input" of the designated type. For example, if the variable is an Integer variable, then the user can enter any whole number. However, the user cannot enter any decimal numbers or any text.

### Variable behavior

The variable displays as a text box where the user can type in their desired value for the variable. In the Web Client only, Integer and Decimal variables also have up and down arrows at the side of the text box to increase or decrease the current input.




*Desktop Client: Example free-input variables*


*Web Client: Example free-input variables*

Invalid entries—for example, attempting to enter a decimal value into an Integer variable—are treated as follows:

- In the Desktop Client, the user can enter any value into the text boxes. However, if the value is invalid, then a validation message displays at the bottom of the dialog. The **OK** button is disabled until the user enters a valid value.
- In the Web Client, invalid values are prevented at the point of entry, so no validation messages display.

**NOTE:** In the Web Client, decimal values for variables are limited to hundredths (two places to the right of the decimal point).

## Variable properties

This section explains how to complete a variable row in the CalcMethodVariables data source when defining a String, Integer, or Decimal variable. Some data source columns do not apply in this case and are not discussed here. If any inapplicable columns are present in the data source, they should be left blank on rows that define free-input variable types. For more information on the CalcMethodVariables data source in general, see Defining calc method variables using a CalcMethodVariables data source.

### General variable properties

All calc method variables use a common set of general properties such as the variable name, display name, and whether the variable is enabled or required. For more information on these properties, see General calc method variable properties.

### Variable-specific properties

| Column Tag | Description |
|---|---|
| `[AutoQuoteString]` | Optional. Specifies whether the string value is placed in single quotation marks when it is written to the calc method location (True/False). If omitted or blank, the default setting is False, which means the string value is not quoted.<br><br>This option is intended to make it easier to create filters based on the selected value, when the selected value is a string and therefore must be wrapped in single quotation marks. For example: `Dept.VP='Smith'`. This option is not commonly used in calc method variables.<br><br>This option applies to String variables but not to Integer or Decimal variables. |

The following properties do not apply to String, Integer, or Decimal variables: PlaceHolderText, ListChoices, ColumnName, AdditionalColumns, ColumnFilter, AllowMultiSelect, DataSourceName, DisplayFormat, Hierarchies, MinDate, MaxDate, TooltipColumn, PrimaryTable, LimitColumn, MinValue, MaxValue, StepFrequency.

## General calc method variable properties

The following variable properties apply to all calc method variable types.

| Column Tag | Description |
|---|---|
| `[Name]` | The name of the variable. This name identifies the variable row in the data source, and is also used as the variable display name to users if a separate display name is not defined in the `[DisplayName]` column.<br><br>The name should not contain any non-alphanumeric characters such as question marks or periods. If you want the variable name that displays to users to include non-alphanumeric characters, use the display name.<br><br>The name cannot be dynamic; it must remain static because it is used to identify the variable. If you are configuring a dependent variable and you need the name to change based on the selection of the parent variable, then you must make the display name dynamic instead of the name. |
| `[DisplayName]` | Optional. The display name of the variable. If defined, the display name will be used instead of the name when the variable displays to users. |
| `[VariableType]` | Specifies the type of variable, such as String, Grid, or ComboBox. The variable type determines the presentation of the user input, as well as the valid values. Most variable types use additional property columns to define the contents and behavior of the variable. For more information, see the topics on each specific type. |

| Column Tag | Description |
| --- | --- |
| [RelativeLocation] | The location to place the variable value when the calc method is inserted, relative within the calc method.<br><br>The location is specified as follows: *<ColumnLetter><CalcMethodRow>*.<br><br>For example:<br>• If the value should be placed in column B of the calc method, within the first row of the calc method, enter the following: B1. If the calc method is a single-row calc method, the calc method row is always 1.<br>• If the value should be placed in column B of the calc method, but within the third row of the calc method, enter the following: B3.<br><br>The entry for the calc method row must be valid within the context of the current calc method. For example, if the current calc method is a three-row calc method, then the only valid entries for the row are 1, 2, and 3. If you attempt to specify B4 for the three-row calc method, the entry is invalid.<br><br>For Calendar variables, the value written to this field is an Excel date serial number. |
| [IsEnabled] | Specifies whether the variable is enabled (True/False).<br>• If True, then the variable will be included in the component.<br>• If blank or False, then the variable will not be included in the component. This evaluation is determined when the component is opened. If the variable is flagged as dependent using the [DependsOn] column, and the current context is a spreadsheet file in the Desktop Client, then the evaluation will be performed again after a value has been selected for the parent variable. |
| [SelectedValue] | For calc method variables, the [SelectedValue] column is only used for dependent variables. When the calc method is inserted and the user selects a value for the parent variable, the selected value is written back to this field temporarily, in the background. This means you can write formulas that change something about the dependent variable based on the selected value of the parent variable. For more information, see Using dependent calc method variables.<br><br>This field cannot be used to set a default value for the variable. Calc method variables do not support the ability to define a default value. |

| Column Tag | Description |
|---|---|
| `[IsRequired]` | Optional. Specifies whether the user must enter a value for this variable (True/False).<br>• If `True`, then the user must specify a value for this variable in order to insert the calc method.<br>• If blank or `False`, then the user can leave the variable blank. The calc method should be configured so that it works as expected if the variable is left blank.<br>The display of required and optional variables depends on the environment. In the Desktop Client, the text **(optional)** follows the name of optional variables. In the Web Client, required variables that do not yet have a selected value are indicated with a red bar along the side of the variable field.<br>**NOTE:** This option does not apply to RelatedColumnValue or GUID variables. |
| `[DependsOn]` | Optional. Specifies that the variable is dependent on a "parent" variable. To make a variable dependent on another variable, enter the name of the parent variable.<br>Some variable types require a parent variable, such as RelatedColumnValue variables. Other variable types can be made dependent as needed.<br>Dependent variables can be updated dynamically in response to the selected value for the parent variable. For more information, see Using dependent calc method variables. |
| `[InsertOnly]` | Specifies when the variable should be displayed:<br>• If `True`, then the variable is only displayed when a user inserts the calc method. It is excluded from the dialog during all other operations, including change (overwrite) and double-click. If all variables for the calc method are set to Insert Only, then the dialog only displays when inserting.<br>• If blank or `False`, then the variable is displayed in all cases.<br>**NOTE:** This option does not apply to RelatedColumnValue or GUID variables. |

## Using dependent calc method variables

When using a CalcMethodVariables data source to define calc method variables, you can make one variable dependent on the selected value of another variable. If the dependent variable is configured with dynamic settings—such as using formulas to determine the list or column or filter—then the

dependent variable can change based on the selected value of the "parent" variable. You can also determine whether the dependent variable displays at all, by making the `[IsEnabled]` property for the variable dynamic.

Dependent variables work as follows:

- When the user specifies a value for the parent variable, that value is placed in the `[SelectedValue]` cell for that variable within the CalcMethodVariables data source, and the file is calculated.
- The settings for the dependent variable are re-read, and the Calc Method Variables dialog is updated for any changes.
- When all variable values are selected, the calc method is inserted and variable values are placed in their designated locations within the calc method. The [SelectedValue] cell is cleared as part of this process, as it is not intended to store values for calc method variables in this location. It is a temporary location, so that dependent formulas can be calculated based on the selected value for the parent variable.

  **NOTE:** There is a current known issue that prevents the `[SelectedValue]` column from being cleared after the insertion occurs. This issue is intended to be fixed in a later patch or release. It is not intended to persist values in the `[SelectedValue]` column.

**IMPORTANT:** The dependent variable behavior only applies when inserting calc methods within the Desktop Client. The Web Client does not currently support dependent variables, except for purposes of configuring RelatedColumnValue variables.

## Configuring a variable as dependent

If you want a variable to be dependent on another variable, set the `[DependsOn]` property for the dependent variable to the name of the "parent" variable. For example, if the Account variable is dependent on the value selected for the Category variable, then enter `Category` in the `[DependsOn]` property for the Account variable.

Once a variable has been designated as dependent on another variable, then you can use formulas in its variable settings to dynamically change the variable based on the parent variable selection. For example, you could use formulas to:

- Specify whether the variable is enabled or not (see the visibility discussion for more information)
- Change the list choices for the variable
- Change the Table.Column for the variable
- Change the filter applied to the column
- Change the display name for the dependent variable

**NOTE:** The variable name must remain static because it is used to identify the variable. If you need the name of the dependent variable to change based on the selection of the parent variable, then you must make the display name dynamic instead of the name.

In all of these cases, the formula should look to the `[SelectedValue]` field of the parent variable and change in response. For example, users could specify a numeric value for the parent variable, and then the dependent variable could be enabled or not depending on whether the numeric value was greater than a certain amount. Or users could select an account category from the parent variable, and then the filter for the dependent variable could change based on the parent variable selection—so that only accounts for the selected category are shown.

The calculation and update of the Calc Method Variables dialog only occurs if a variable is flagged as dependent using the `[DependsOn]` property, and only dependent variables are updated. If you use formulas in variable settings but the variable is not flagged as dependent, then the variable will not update once the Calc Method Variables dialog has been presented to the user.

## Using RelatedColumnValue variables

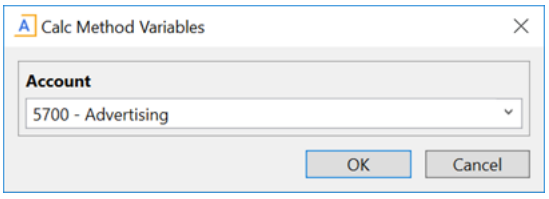
RelatedColumnValue variables can be used to bring in additional column values based on the user's selection for a parent Grid or ComboBox variable. This is a special variable type that does not display to the user in the Calc Method Variables dialog—it is only used to return additional related values into the calc method.

RelatedColumnValue variables must have a specified parent variable using the `[DependsOn]` property. In this case, the purpose of the dependency is only to determine the parent value for which to return the related column values. The RelatedColumnValue variable does not need to be set up with any formulas in order to make this determination.

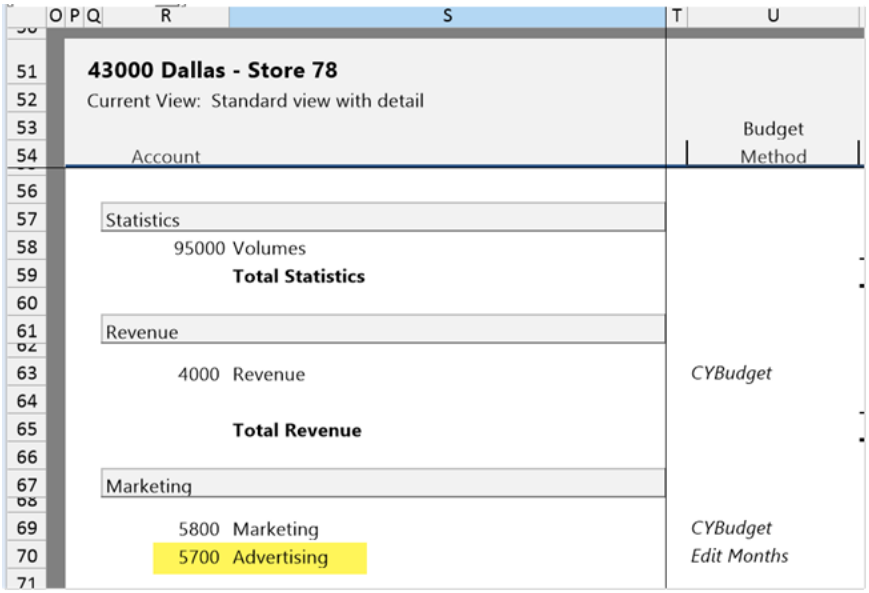
For example, imagine that you are prompting users to select an account to place in the calc method, and you also want to place the account description in the calc method. To do this, you first set up the parent variable to display a list of accounts. Then, you create a RelatedColumnValue variable to return the account description, and configure it as dependent on the parent Account variable.

| | A | B | C | D | E | F | G | H | I | K | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 27 | | | | | | | | | | | |
| 28 | | | [CalcMethodVariables;Budget;Edit Months] | [Name] | [DisplayName] | [VariableType] | [RelativeLocation] | [IsRequired] | [IsEnabled] | [DependsOn] | [ColumnName] |
| 29 | | | [Variable] | Acct | Account | ComboBox | R1 | TRUE | TRUE | | acct.acct |
| 30 | | | [Variable] | Desc | | RelatedColumnValue | S1 | | TRUE | Acct | acct.description |

When the user inserts this calc method and the Calc Method Variable dialog displays, only the Account variable is visible.



When the calc method is inserted, Axiom finds the specified column for the RelatedColumnValue variable—in this example, Acct.Description—and returns the value for the selected account. Both values are then placed into the specified relative locations within the calc method.




The Related Column Value variable provides an alternative to using GetData functions to return these associated values. Avoiding use of unnecessary GetData functions can improve file performance.

## Visibility of dependent variables in the Calc Method Variables dialog

When setting up dependent variables, you should consider whether you want the dependent variable to be visible in the Calc Method Variables dialog from the start, or whether it should be hidden initially. Dependent variables behave as follows, based on the value of the `[IsEnabled]` property:

- If the dependent variable is enabled by default, then the dependent variable displays in the Calc Method Variables dialog when it opens, along with the parent variable. In this configuration, the user is not required to select a value for the parent variable before selecting a value for the child variable. You should make sure that the parent variable is configured as required, so that the user cannot skip the parent variable entirely.
- If the dependent variable is not enabled by default, then the dependent variable is omitted from the Calc Method Variables dialog when it opens. Once the user selects a value for the parent variable, the dependent variable may or may not become visible in the dialog, depending on the formula that was used in the enabled setting. This configuration ensures that the user cannot select a value for the child variable until they have first selected a value for the parent variable.

  For example, the formula might be set up so that once any value is set for the parent variable, the dependent variable becomes enabled. Or the formula might be set up so that the dependent variable only becomes enabled if the parent variable value meets a certain criteria.

**NOTE:** RelatedColumnValue variables never display to users. For these variables, the `[IsEnabled]` property simply determines whether the RelatedColumnValue variable is processed or not when the parent variable value is selected.

## Defining calc method variables within the calc method properties (legacy approach)

When defining variables for a calc method, you can define them within the calc method properties. This is the "legacy" approach to defining calc method variables. When using this approach, the variables are stored within the centralized calc method library along with the actual rows of the calc methods. All files with access to the calc method library also have access to the variables.

Legacy calc method variables support the following variable types:

- **String**: The user can enter any string value.
- **Integer**: The user can enter any whole number.
- **Decimal**: The user can enter any numeric value, including decimals.
- **List**: The user can select any item in a defined list.
- **Grid**: The user can select any item from a specified table column (for example, ACCT.ACCT to select from a list of accounts).
- **Related Column Value**: This variable type is not presented to the user. It is used to return a related column value for a parent Grid variable, so that the value can be referenced within the calc method.
- **GUID**: This variable type is not presented to the user. It generates a globally unique identifier (GUID) and places it into the designated location when the calc method is inserted.

Other than related column values, legacy calc method variables do not support any other type of dependency between variables. Additionally, legacy calc method variables have a limited ability to read information from the file where they are to be inserted, using cell references. For example, when setting up a Grid variable, you can use a cell reference to read a filter from the file, to filter the list of values in the column.

**To define legacy variables for a calc method:**

You can define legacy variables when you are creating a new calc method or editing an existing calc method. The following procedure assumes that you are editing an existing calc method. The same user interface is available when you are creating a new calc method.

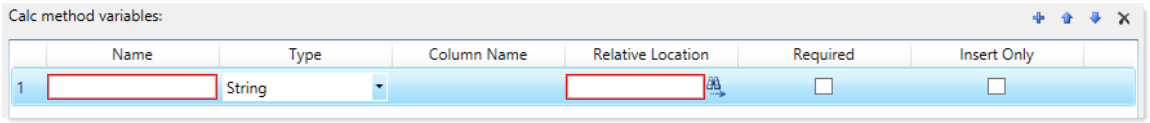
1. Optional. Insert the calc method into the sheet and then select the relevant rows (or select an instance of the calc method that is already in the sheet).

This step is not required, but having the calc method selected in the sheet allows Axiom to show you exactly where the variable value will be inserted into the calc method. (When creating a new calc method, you would have selected the applicable rows anyway as part of the creation process.)

2. On the **Axiom** tab, in the **Advanced** group, select **CM Library > Manage Calc Methods**.
3. In the **Manage Calc Methods** dialog, select the calc method and then click **Edit**.

   The **Edit Calc Method** dialog opens, displaying the settings for the selected calc method.
4. In the **Calc method variables** section, click **Add variable** ✚.

   A new variable row is added to the grid.

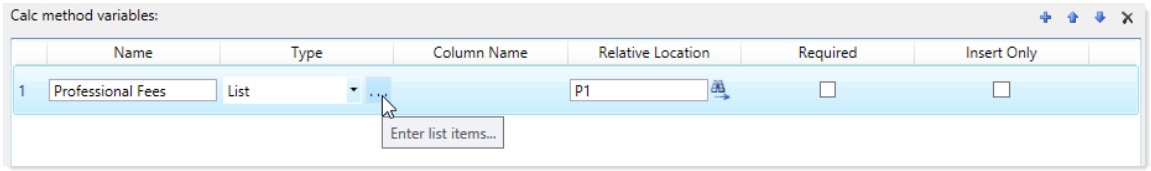



5. In the **Name** box, type a name for the variable. A cell reference can also be used, enclosed in brackets—for example, `[A5]`.

   The name displays in the Calc Method Variables dialog, and tells the user what value is being specified. The name can be just a term ("Account") or it can be an instructive phrase ("Select an account").
6. In the **Type** box, select the type of variable.
   - **String**: The user can enter any string value.
   - **Integer**: The user can enter any whole number.
   - **Decimal**: The user can enter any numeric value, including decimals.
   - **List**: The user can select any item in a defined list. List values are presented to the user in a drop-down list.
   - **Grid**: The user can select any item from a specified table column (for example, ACCT.ACCT to select from a list of accounts). Column values are presented in the **Choose Value** dialog, using a searchable grid.
   - **Related Column Value**: This variable type is not presented to the user. It is used to return a related column value for a parent Grid variable, so that the value can be referenced within the calc method. The Related Column Value variable must be placed underneath its parent Grid variable.
   - **GUID**: This variable type is not presented to the user. It generates a globally unique identifier (GUID) and places it into the designated location when the calc method is inserted.
7. If the variable type is **List** or **Grid**, define the list of allowed variable values.

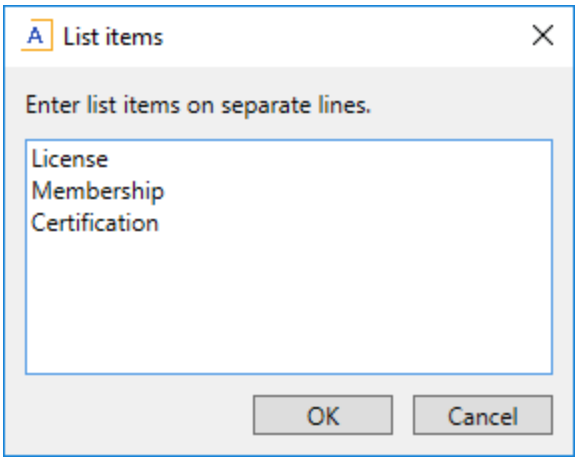
### List: Define the list values

To define the list of allowed values for a List variable, click the **[...]** button to the right of the type selection.




In the **List Items** dialog, type each item on a separate row. To create a new row, use the `Enter` key.

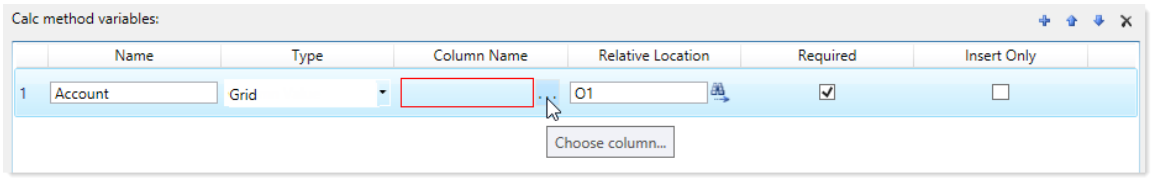
Cell references can also be used to define list items. Each cell reference must be placed on its own row, enclosed in brackets.




*Example list definition*

### Grid: Select the source column

To specify the column to use for the Grid variable, click the **[...]** button in the Column Name field.



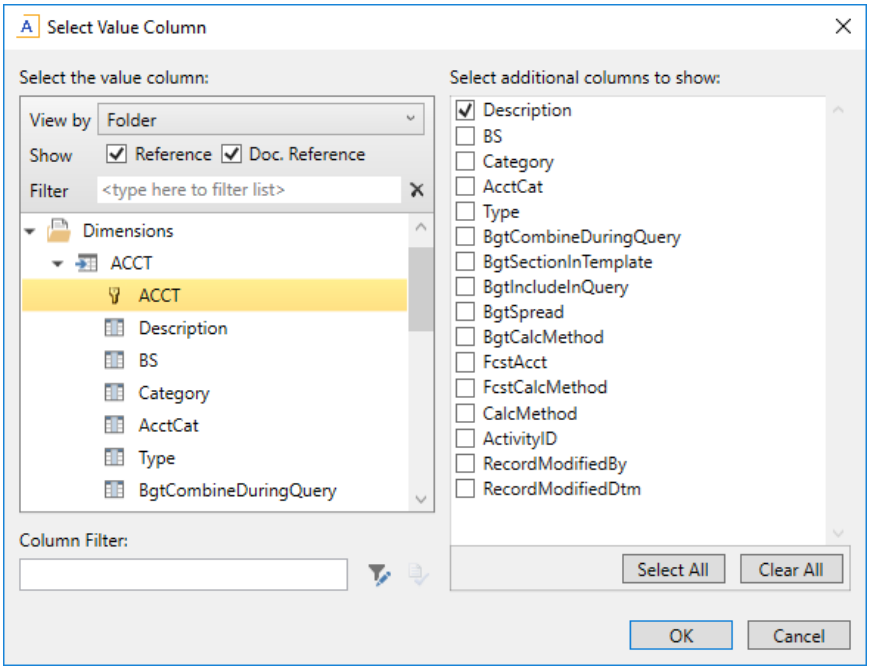

The **Select Value Column** dialog opens.

- In the left-hand side of the dialog, select the column. You can change the table view and filter the list to find the appropriate column. Only columns from reference tables or document reference tables can be used.

By default, the user can select any value in this column. If desired, you can define a filter to limit the list of values. In the **Column Filter** box, type a filter criteria statement, or click the **Filter Wizard** button to create one. A cell reference can also be used, enclosed in brackets—for example, `[A5]`.

- In the right-hand side of the dialog, select any additional columns that you want to present to users when they select a value.

  **NOTE:** This option is only available when the selected column is a key column. If the column is a grouping column, then no additional columns can be shown. When the user selects a value, the list will be the list of unique values in the grouping column.




*Example column selection*

8. In the **Relative Location** box, enter the location where the variable value will be placed, relative within the calc method.

   The location is specified as follows: `<ColumnLetter><CalcMethodRow>`.

   For example:

   - If the value should be placed in column B of the calc method, within the first row of the calc method, enter the following: `B1`. If the calc method is a single-row calc method, the calc method row is always 1.
   - If the value should be placed in column B of the calc method, but within the third row of the calc method, enter the following: `B3`.

   The entry for the calc method row must be valid within the context of the current calc method. For example, if the current calc method is a three-row calc method, then the only valid entries for the row are 1, 2, and 3. If you attempt to specify B4 for the three-row calc method, the entry is invalid.

   If the calc method rows are selected in the sheet (as described in step 1), then you can click **Show location within the current spreadsheet selection**  , and the cursor will move to the location where the value would be placed.

9. If you want the user to be required to specify a value for the variable, select the **Required** check box.

   If the variable is required, then users must specify a value in order to insert the calc method or use it to overwrite another.

   If the variable is not required, then users can leave the variable blank, resulting in the corresponding cell being blank when the calc method is inserted. If the calc method is overwriting another calc method, the original cell contents are retained if the variable is left blank.

   You cannot specify a default value for a variable; either the user specifies a value, or the cell is left blank.

10. If the variable should only display when the calc method is inserted by a user, select the **Insert Only** check box.

    Some variables, such as selecting an account for the row, should only be displayed when the calc method is inserted. When a user changes an existing row using the calc method, the existing account should be retained.

    If Insert Only is selected, then the variable will not display when a user performs an overwrite with the calc method, or when a user double-clicks an existing row in the sheet to edit the variable values. If Insert Only is not selected, then the variable will display in all cases.

    If all of the variables for the calc method are set to Insert Only, then the Calc Method Variables dialog will only display when the calc method is inserted.

11. Repeat steps 4-10 to define any additional variables for the calc method.

When the Calc Method Variables dialog is displayed to the user, variables are listed in the order that they are listed in this grid. You can change the order of variables by selecting a variable row and then clicking the **Move variable up**arrow or the **Move variable down** arrow.

**NOTE:** The presence of a Related Column Value variable restricts how variables can be moved up and down, because the Related Column Value variable must be underneath its parent Grid variable. If you have a Grid / Related Column Value pair, and you want to change their order in the list of variables, then you must move the other variables up or down in the list instead of attempting to move the pair.

If you want to delete a variable, select the variable row in the grid and then click **Remove variable** ✕.

12. Click **Apply** to save your changes to the calc method (or **OK** if you are finished editing the calc method).

## Sorting behavior for column values

If the variable uses a column value, the list of values is sorted based on the source column. In some cases the values may not be displayed as expected:

- If the column is from a driver table, the values will most likely not be displayed in the order they are stored in the driver file. When the values are saved to the database from the driver file, they are sorted based on the key column of the table (column A of the driver sheet).
- If the column is a string column that contains numeric values (values that are all numbers and values that start with numbers), Axiom will attempt to sort these values in numeric order. Values that start with letters will then be sorted in alphabetical order.

## Using cell references with legacy calc method variables

Several properties of legacy calc method variables support use of cell references. Instead of "hard-coding" a value within the calc method properties, you can point to a cell location in the file. This cell location can contain regular text, or it can use a formula to dynamically derive a value, or it can contain a range that Axiom will use to dynamically generate a list of choices.

The following properties support cell references:

- The variable name
- If the variable type is List, the list items
- If the variable type is Column Value, the column filter

To use a cell reference, enter the reference into the property like so:

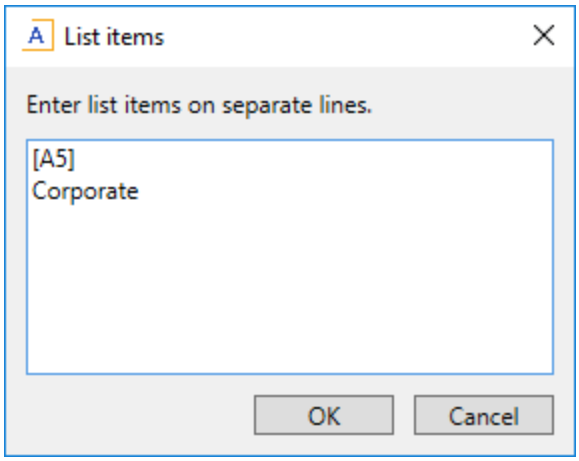
```
[A5]
```

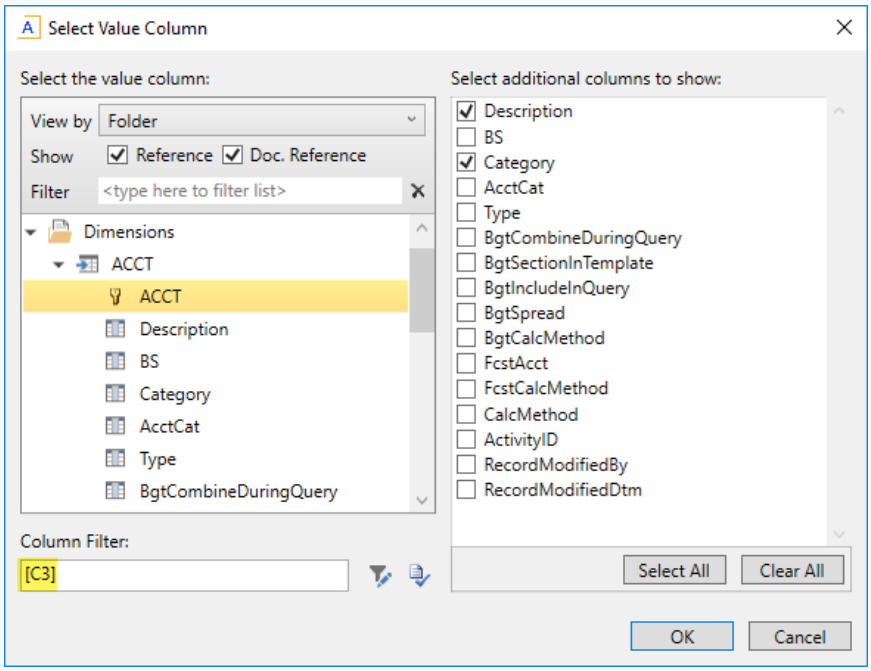
or

```
[Sheetname!A5]
```

Cell references are always absolute. An entry of `[A5]` always points to cell A5, regardless of the placement of the inserted calc method or the target cell for the variable. However, you can use a formula in cell A5 to dynamically change the contents—for example, depending on the current insertion point location. The workbook is calculated before the Calc Method Variables dialog opens.

For list items only, cell ranges can also be used.

| Variable Property | Cell Reference Support |
|---|---|
| Variable name (All variable types) | Enter the cell reference into the **Name** field. When the Calc Method Variables dialog is presented to the user, the name of the variable will be derived from the referenced cell. |
| List items (List only) | Enter the cell reference into the **List items** dialog on its own line, just as you would if it was a hard-coded value. The list can contain multiple cell references and/or values, the results of which will all be joined to create the final list.<br><br>For list items only, the referenced cell can contain a range, such as `List!B7:B27` (without brackets). The range can be hard-coded within the cell as text, or it can be the result of a formula. In this case, Axiom takes all values in the range, and adds them to the list as individual choices. The range must be one dimensional—meaning either one column wide and several rows tall, or several columns wide and one row tall. If the range is a block, it is ignored.<br><br>If desired, you can also enter the range directly within the **List items** dialog. In this case, use brackets: `[List!B7:B27]`.<br><br>  <br>*Example list items with cell reference*<br><br>In this example, the contents of cell A5 will be combined with the hard-coded item "Corporate" to create the full list of choices. If cell A5 contains a range, then the values in that range are added to the list. |

| Variable Property | Cell Reference Support |
|---|---|
| Column filter (Grid only) | Enter the cell reference into the **Column Filter** box when selecting the column to use for the list. The referenced cell must contain a full, valid filter.<br><br>If you are using a cell reference for the column filter, the only entry in the **Column Filter** box must be the cell reference. For example, you *cannot* use a "mixed" configuration where part of the filter is defined in the box and the remaining part is read from the cell reference.<br><br>  <br><br>*Example column filter with cell reference*<br><br>In this example, the column filter is being read from cell C3. If cell C3 contains `Acct.Category='Revenue'`, then when the Calc Method Variables dialog is presented to the user, the account list will be filtered to only show revenue accounts. |

## Allowing users to edit values for calc method variables on double-click

When a calc method uses variables, the user specifies values for the variables when inserting the calc method, and those values are placed in the designated location within the calc method. If the user later wants to change the values used for the variables, by default the only way to do so is to edit the corresponding cells in the spreadsheet.

However, you may want to configure these cells as locked in the sheet, to prevent users from accidentally clearing necessary values or entering invalid codes. In this case, you can still give users the option of editing the variable values by enabling double-click actions for the sheet.

If double-click is enabled, a user can double-click on a calc method in the sheet in order to re-open the calc method variables dialog. The dialog shows the current variable values used in the calc method, for any variable that is not configured as insert-only. The user can change any of these values and click OK, and the calc method in the sheet will be updated for the new values.

Enabling this behavior requires the following:

- The template that the plan file was built from must be set up for template validation. See Template validation.
- The **Axiom Double-Click** setting must be enabled for the sheet on the Control Sheet. This setting is located in the **Sheet Options**.
- For all calc method variables that display to users, at least one of these variables must *not* have the **Insert Only** option checked. If all visible variables have Insert Only enabled, then they only apply during calc method insert and will not display on double-click.

This behavior only applies when using standard spreadsheet plan files. Form-enabled plan files do not support the ability to reopen the calc method variables dialog on double-click.

**NOTES:**

- Although Related Column Value variables do not display in the calc method variables dialog, the values for these variables will also be updated if the parent variable value is changed using double-click editing.
- GUID variable values are not changed when using double-click editing.

# Setting up calc method controls for a sheet

When designing a template sheet (or a sheet in any file that supports use of calc method libraries), you can use calc method controls to specify where calc methods can be used in the sheet. Using calc method controls, you can:

- Control exactly where in a sheet that a user can insert or change a calc method. Each row can be marked to allow insertion, overwriting, or both.
- Specify the calc methods that can be used to insert or change on a particular row. Calc methods can be specified by individual calc method name and by calc method group.
- Define custom insertion points, to guide users to the appropriate areas for calc method insertion. Custom insertion points display on the **Add Row(s)** menu; when one is selected, the user is taken directly to the appropriate row.

To configure calc method controls for a sheet, you must define a calc method control column on the sheet, and then mark each row that you want enabled for insertion or overwriting. You can create these tags manually, or use the user interface (available on the right-click menu) to insert or edit the tags.

Calc method controls tag summary

| Tag Type | Tag Syntax |
|---|---|
| Primary tag | `[InsertCMColumn; HeaderLabel]` |
| Row tags | `[InsertCM; InsertionPointLabel; CalcMethodList; GroupList; AllowedAction; InsertLocation; MaxInsertCount; DynamicMarker]` |

Calc method controls are optional. If no calc method control column is present in a sheet, then calc methods can be inserted or overwritten anywhere on the sheet. However, if a sheet has a calc method control column, then calc methods can *only* be inserted or overwritten on rows that are marked as enabled.

If a row is marked as enabled, users can only insert or change a calc method if they have security rights to do so. For example, if a row is enabled for overwriting, but the user does not have security rights to overwrite calc methods for the file group, then the change action will not be available to the user.

**NOTES:**

- Axiom supports an alternate method of controlling calc method insertion, known as dynamic insertion controls. When using this method, the user selects an account (or other dimension item) that they want to work with, and the dynamic insertion controls do the work of determining the correct location and then either automatically inserting a calc method or taking the user to an existing row in the sheet. This method can be more user-friendly, but also requires more work for the template designer to set up the appropriate spreadsheet logic. For more information, see Using dynamic insertion controls.
- Administrators and users with the **Manage Calc Methods** security permission can always insert or change calc methods anywhere in a sheet, even if calc method controls are being used. There is no "override" mechanism—the controls are simply ignored for these users. However, note that if you insert a calc method by selecting a custom insertion point, you are always limited to the calc methods allowed for insert at that location. If you want to insert a different calc method at that location and you have the security rights to do so, then you must insert it manually (by placing your cursor at the insert location and selecting **Add Row(s) > Insert Calc Method**, instead of using the custom insertion point).

## Defining a calc method control column

The calc method control column (InsertCMColumn) serves two purposes:

- It enables calc method controls for the sheet.
- It holds the row tags to mark individual rows as enabled for calc method insertion or overwriting.

Typically the calc method control column is located to the far right of the sheet, past any data area that users work with or view. However, it can be placed wherever you like, within the first 500 rows.

You can enter the control column tag into the sheet manually, or you can use the insertion tool:

1. Right-click in the desired cell, and then click **Calc Methods > Insert Column Control Tag**.

   The **Insert Column Control Tag** dialog opens.

2. In the **Menu label** box, type the text that you want this set of insertion points to display under on the **Custom Inserts** menu.

   This text serves as the "header" for each individual custom insertion point defined in the column. This text could be sheet related ("Add rows to Budget sheet") or it could be related to the type of calc method actions enabled in the column ("Add new employees").

   If you do not want this set of calc method controls to appear on the menu, then clear the **Show in Custom Insert Menu** check box. Generally, the only use case for this is if the column only controls change actions, and therefore no items would appear on the menu anyway. If the column allows inserting calc methods, then menu text should be defined to guide users to insertion points.

3. Click **OK** to insert the tag and enable calc method controls for the sheet.

The tag is inserted into the current cell as follows: `[InsertCMColumn;HeaderLabel]`

There is no associated user interface for editing column control tags once they have been inserted. If you later want to change or remove the header text for an InsertCMColumn, make the edit directly within the cell. For more information on the tag syntax, see Calc method control tag syntax.

Once the InsertCMColumn tag has been placed in a sheet, then calc method actions are only allowed on rows that are marked as enabled. If no rows are marked as enabled, then users cannot insert or change calc methods anywhere in the sheet.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

## Marking rows as enabled for insertion and/or changing

Each row that you want enabled for calc method insertion or changing (overwriting) must be marked with an InsertCM tag.

Tags that allow calc method insertion are typically placed at the bottom of a section in a sheet, beneath any "hard-coded" rows, or beneath rows that will be brought in via Axiom query when plan files are built. In this case the tags are placed directly in the template sheet.

However, insertion can be allowed anywhere. For example, you may have a calc method that uses detail sub-rows to calculate a total, and you want to allow users to insert additional detail rows. In this case the tags must be saved as part of the calc method, so that the tags are brought in with the calc method when the plan file is built or when the calc method itself is inserted.

Tags that allow overwriting must be saved within the calc method that you want users to be able to change. If the calc method is a multiple-row calc method, only the first row must contain the InsertCM tag to allow the calc method to be changed.

The InsertCM tag has multiple parameters and can be fairly long. While you can manually type the tag within the sheet, the easiest method is to use the insertion tool to create the tag:

1. Place your cursor in the row that you want to enable for insertion and/or overwriting, within the calc method control column (the column marked with InsertCMColumn). Right-click in the cell and then click **Calc Methods > Insert Row Control Tag**.

   The **Insert Row Control Tag** dialog opens.

   Remember that some tags must be saved within calc methods instead of within the template itself. You would need to insert the calc method into the sheet, add the tag to the appropriate column, and then save the modified calc method back to the library.

2. If this row will be enabled for insertion, type the desired text for the **Custom Inserts** menu into the **Menu label** box.

   For example, if this insertion point is where users can add new hourly employees, the menu label could be "Add New Hourly Employee".

   If the row will only allow overwriting, then this setting does not apply, because overwriting locations are not displayed on the menu. In this case, clear the **Show in Custom Insert menu** check box.

   This setting is optional in any circumstance, however, if the row allows insertion then menu text should be defined to guide users to the insertion point.

3. For **Actions to allow on this row**, select the desired calc method action:

   - **Insert**: Users can insert calc methods at this location.
   - **Change**: Users can change the calc method that starts with this row.
   - **Insert/Change**: Users can insert or change calc methods at this location.

   **NOTE:** A fourth option is available, named **GoTo**. This option is only for use with dynamic insertion controls. If you select this option for use with standard calc method controls, then the row will be ignored.

4. Select one of the following to determine which calc methods can be used at this location:
    - **Allow any calc method** (default)
    - **Allow only the selected calc methods and groups**

    If you selected **Allow only the selected calc methods and groups**, then select the check box for each individual calc method and/or calc method group that you want to allow at this location. If you select a group, then all calc methods in that group can be used at this location.

5. Optional. Configure **Max insert count** as appropriate.

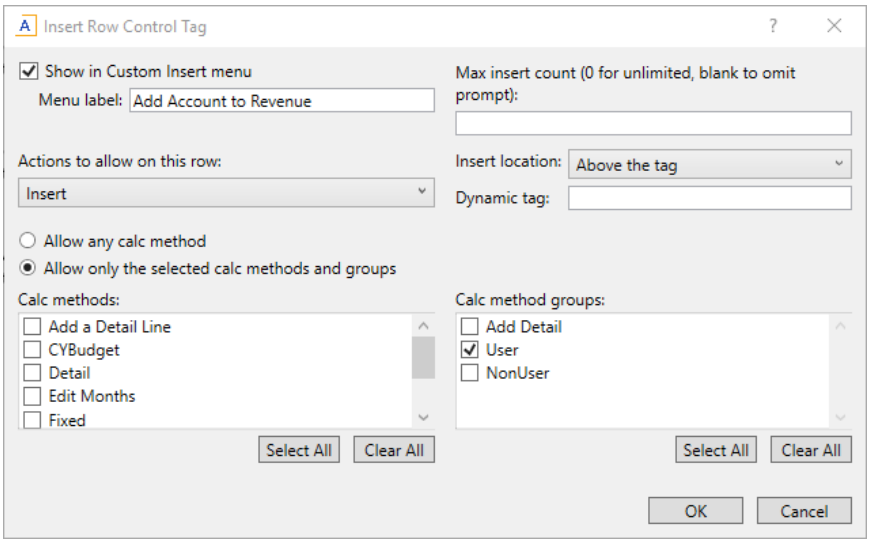
    This setting controls how many instances of the selected calc method the user can insert. For example, if the user is inserting a detail row, you may want to allow the user to insert more than one detail row at a time. You should evaluate this setting depending on whether only one calc method is eligible for insert at this location, or whether multiple calc methods are eligible for insert:

    - If the **Insert Calc Method** dialog will display to the user (i.e. more than one calc method is available for insert at this location), then this setting controls what the user can enter into the **Number of items to insert** field in the dialog. In this circumstance, blank or 0 means unlimited—the user can insert as many instances as they wish. You can enter a number if you wish to limit the number of instances that can be inserted, or you can enter 1 if you do not want the user to be able to insert more than one instance of the calc method.
    - If only one calc method is available for insert at this location, then this setting controls whether or not the user will be prompted to insert more than one instance of the calc method, and if so what is the maximum number they can input. You should leave this setting blank if you do not want the user to be prompted. If you want the user to be prompted, you can enter 0 for unlimited instances, or enter a number higher than 1. (Entering 1 does not make sense in this circumstance, because this will cause the prompt to display but the user will be unable to change it to a number higher than 1.)

6. Optional. For **Insert location**, specify where the calc method should be inserted, relative to the current row:
    - **Above the tag** (default): The calc method is inserted directly above the current row.
    - **Below the tag**: The calc method is inserted directly below the current row.

7. Click **OK** to insert the tag and enable the row for insertion and/or overwriting.

The following screenshot shows an example Insert Row Control Tag dialog:




The selections in this example result in the following tag:

```
[InsertCM;Add Account to Revenue;;Budget;Insert;Above]
```

**NOTE:** The **Dynamic tag** option within the dialog is only for use with dynamic insertion controls. It is ignored when using standard calc method controls. Whether it is blank or has text, it has no impact on the row.

Once the tag has been inserted into a cell, you can edit it manually, or you can use the right-click menu: **Calc Methods > Edit Row Control Tag**. For more information on the control tag syntax, see Calc method control tag syntax. This section also has example InsertCM tags for various scenarios.

## Using multiple control columns in a single sheet

You can have multiple calc method control columns defined on a sheet. For example, if you have several departments in a plan file, users may need to add departments or add accounts. You could have two separate calc method insertion columns as follows: `[InsertCMColumn;Insert Dept in Budget Sheet]` and `[InsertCMColumn;Insert Acct in Budget Sheet]`.

If a sheet has multiple calc method control columns, and the same row is marked as enabled within multiple control columns, then the settings of the individual control columns are combined to result in the allowed actions and the allowed calc methods. This means:

- If one tag specifies Insert, and another tag specifies Change (on the same row), then users can insert and change on that row.
- If a user selects a custom insertion point, the calc methods are limited to the items allowed by the InsertCM tag that defines the insertion point. However, if a user chooses to manually insert a calc method, the user has access to the full set of calc methods allowed for that row (in all tags).
- If a user changes a calc method, the user has access to the full set of calc methods allowed for that row (in all tags).

## Using dynamic insertion controls

In addition to the standard calc method controls, Axiom supports an alternative method that allows you to dynamically determine where a new calc method should be placed in a sheet based on a user's dimension selection (such as an account), and then automatically insert it.

For example, imagine that the user wants to insert a Travel account. Using the standard calc method controls, the user must either manually navigate to the appropriate area of the sheet, or select the appropriate custom insertion point from the **Add Rows** menu. Depending on how many insertion points are available, this may be a lot of items to choose from. Additionally, maybe the account is already in the sheet, and the user doesn't know it.

Using the dynamic insertion controls, the user experience can instead be something like this:

- The user selects to "Update an account" (or some similar text) from the **Add Rows** menu. It does not matter what type of insertion the user wants to make (for example, a brand new account, or new detail rows for an existing account), or where the rows need to go—it can all be performed using this single option.
- A dialog box presents the user with a list of accounts (or any desired dimension). The account list can be filtered so that the user only has to choose from accounts that are relevant to the plan.
- Once the user selects an account, the account is placed in a designated location in the sheet and then the sheet is calculated. This process is transparent to the user. The assumption is that the plan files have been designed so that based on the user's account selection, the appropriate row for calc method insertion is now identified.
- A calc method is inserted in the correct location on the sheet, using the preferred calc method for the selected account. Optionally, you can allow the user to select which calc method they want to use, and/or you can allow the user to specify how many items to insert (for example, when inserting detail rows).
- If the selected account already exists in the sheet and no available actions can be taken for that account (such as inserting additional detail rows), then the user is simply taken to the appropriate row in the plan file.

It is important to understand that the dynamic controls alone do not determine where the account should be placed in the sheet (or whether the account is already there). The dynamic controls enable the dimension selection for the user, stamp the user's selection in the sheet, and then calculate the sheet before performing the calc method insertion. It is up to the template designer to create the spreadsheet logic that identifies the appropriate location for calc method insertion (or navigation to an existing row).

Dynamic insertion controls can provide a very intuitive and streamlined experience for the end user, but they are more complex for the template designer to set up and may require advanced Excel knowledge to create the necessary spreadsheet logic.

Dynamic insertion controls use a special control column of DynamicCMColumn. This column is similar to the standard InsertCMColumn, but it uses additional parameters to enable the dynamic insert. Rows are flagged using the same InsertCM tags, however, certain settings are ignored in this context, and a few additional options are available.

Dynamic insertion controls tag summary

| Tag Type | Tag Syntax |
|---|---|
| Primary tag | `[DynamicCMColumn; Label; DynamicMarker; Table\|Filter\|SheetDestination\|CalcMethodDestination; MultipleInsert]` |
| Row tags | `[InsertCM; InsertionPointLabel; CalcMethodList; GroupList; AllowedAction; InsertLocation; MaxInsertCount; DynamicMarker]`<br><br>This is the same syntax used with the standard calc method controls (InsertCMColumn). |

**NOTES:**

- By default, dynamic insertion controls find a single match and then insert or go to that location. If multiple matches exist, the first one is used. You can configure the controls to allow multiple insertions, but certain behavior restrictions apply.
- When using DynamicCMColumn, the **Insert Calc Method** option does not appear on the **Add Row(s)** menu, even if the user's cursor is on a row that is enabled for insert. All calc method insertions are controlled by use of the dynamic insertion menu item. (If necessary, you can set up an InsertCMColumn in the same sheet, and enable the menu item that way.)
- For administrators and users with the **Manage Calc Methods** security permission, the behavior of dynamic insertion controls is the same as for non-admin users. However, as always, calc method administrators have the right to insert or change calc methods anywhere in the sheet, regardless of any calc method control tags. The dynamic insertion behavior will only apply if they select the relevant custom insert from the **Add Row(s) > Custom Inserts** menu.

## Defining the dynamic insertion control column

To configure dynamic insertion controls for a sheet, you must specify a dynamic insertion control column (DynamicCMColumn). This column is similar to the standard InsertCMColumn, but uses different syntax to enable the dimension selection and the placement of values on the sheet.

To define the DynamicCMColumn, place the following within any cell in the first 500 rows of the sheet:

`[DynamicCMColumn; Label; DynamicMarker; Table|Filter|SheetDestination|CalcMethodDestination; MultipleInsert]`

For full details on the syntax for the dynamic insertion control column, see Calc method control tag syntax.

The DynamicCMColumn tag must be manually typed in the cell. No helper dialog is available to build the tag.

### Configuring the InsertCM tags for dynamic insertion

Within the dynamic insertion control column, rows are marked using the same InsertCM syntax used with the standard calc method controls. For more information on InsertCM syntax, see Calc method control tag syntax. Although the syntax is the same, some behavior differences apply.

When using dynamic insertion controls, the goal is to design the sheet so that once the user's dimension selection is placed in the sheet and the sheet is calculated, only one row is marked as "active" for the dynamic insertion. This can be accomplished in one of two ways:

- The InsertCM tags can contain a text "marker" that matches the dynamic marker defined in the DynamicCMColumn tag. The first row that contains this marker is the active row.

  The dynamic marker is placed in the last parameter of the InsertCM tag, and is only valid for use within a dynamic insertion control column. If used within a standard calc method control column (InsertCMColumn), it is ignored.

- If no marker is defined in the DynamicCMColumn tag, then the active row is the first row that contains an InsertCM tag.

In both cases, it is assumed that the InsertCM tags are constructed using a formula. The formula should be designed so that based on the user's dimension selection, only one InsertCM tag in the column is now flagged with the dynamic marker or visible within the sheet (depending on which approach you are using). How this formula is constructed and what spreadsheet logic is used to determine the active row is entirely up to the template designer.

The following additional behavior differences apply when using InsertCM tags in a dynamic insertion control column:

- The InsertionPointLabel text is ignored. When using a dynamic insertion control column, the only text that displays on the menu for the column is the label defined in the DynamicCMColumn tag.
- The AllowedAction parameter supports an additional action of `GoTo`. If GoTo is specified as the action, then when the current row is identified as the active row, the user is taken to the current row but no calc method insertion occurs. The GoTo action is intended to be used in situations where the user's selected dimension item already exists in the sheet. In that case, the user should be taken to the existing row rather than inserting a new row. For more details, see the following section *Using dynamic insertion with existing rows*.

To create InsertCM tags, you can manually type the tag or use the insertion tool on the right-click menu (**Calc Methods > Insert Row Control Tag**). In most cases it will be easier to start the tag using the insertion tool, and then convert it to a formula to accommodate the necessary conditional logic. Once the tag is converted to a formula, you will no longer be able to use the insertion tool to edit the existing tag.

### Using dynamic insertion with existing rows

When using dynamic insertion controls, the user is first prompted to choose an account (or whichever table is specified in the DynamicCMColumn tag). It is possible for the user to choose an account that already exists in the sheet. The spreadsheet logic used in the sheet to determine the active row must address this situation.

For example, imagine that the user selects the Travel account, and that account is already in the sheet.

- If the calc method used for the Travel account allows for detail rows, then you probably want to configure the dynamic insertion so that the user can insert more detail rows. In this case the spreadsheet logic would need to identify one of the existing detail rows as the active row, and the InsertCM tag would need to allow for insertion of the detail calc method. You could also give the user the option of inserting multiple detail rows.
- If the calc method used for the Travel account does not allow for detail rows, then the dynamic insertion should be configured so that the user is simply taken to the existing row. To accommodate this, the InsertCM tag now supports an additional action of GoTo (instead of Insert and/or Change) for the AllowedAction parameter. If the active row is flagged with GoTo, then the user is taken to the active row and no further action occurs. The user can then edit the existing row as desired.

**TIP:** If you want the user to be able to change the existing row by applying a different calc method, then you can use Change as the AllowedAction instead of GoTo. For more details, see the following section *Changing calc methods when using dynamic insertion*.

It is also possible to configure the sheet so that if the user selects an account that is already in the sheet, no row becomes the active row. In this case, a message displays to the user as follows: "There is nothing to insert in the sheet for <the selected item>." However, this is not a recommended configuration, since it does not let the user know that the account is already in the sheet.

Whichever approach is used, the spreadsheet logic should ensure that a duplicate row is not inserted into the sheet for the selected account (unless that edge case behavior is explicitly intended by the plan file design).

### Allowing users to change calc methods when using dynamic insertion

If you want users to be able to change calc methods on existing rows in the plan file, you can configure the dynamic insertion controls to allow this. The following design considerations apply:

- On existing rows, InsertCM tags can use Change as the allowed action instead of GoTo. In this case the user will be taken to the row just like when using GoTo, but if the user has **Allow Calc Method Change** permissions then they can use the **Add Row(s) > Change Calc Method** menu option to select a different calc method. Use of GoTo explicitly denies use of Change on the row (unless the user is a calc method administrator and therefore can change calc methods anywhere).

- To flag active rows, you must use the dynamic marker method. This way, all InsertCM tags can be visible in the sheet at all times, so that if the row is configured to allow overwrite then the user can do so. The formulas that determine the InsertCM tags just need to identify which row should be flagged with the dynamic marker at any one time.

Alternatively, you could set up a separate standard InsertCMColumn in the sheet, and use that column to control access to changing calc methods.

### Allowing multiple insertion points

You can use the optional MultipleInsert parameter to allow the dynamic insertion process to insert at multiple locations instead of just one. If this parameter is used, then a calc method will be inserted at all matching locations, instead of just the first matching location. This behavior applies regardless of whether matched rows are determined using the dynamic marker text or just by hiding and showing the InsertCM tag as appropriate.

This behavior should only be used for configurations where multiple insertions of the same item make sense. For example, in a traditional single-department budget you would not want to allow multiple insertions of the same account within a single sheet. However if the plan file was for multiple departments, then you could have multiple insertions of the same account, one for each individual department.

Use of this feature has several limitations as compared to the standard single-insertion behavior:

- All matching rows must be limited to a single calc method. This means that the InsertCM tag for the matching row must either specify a single allowed calc method, or a calc method group that only contains one calc method. Also, the MaxInsertCount parameter for the InsertCM tag must be blank (meaning inserting multiple instances of the calc method at a single location is not allowed).

  **NOTE:** Each matching row can allow a different calc method—so Insert Monthly might be inserted at one matching location, and then Spread Total might be inserted at a different matching location. All matching rows do not need to be limited to the same calc method.

- The allowed calc method for the matching row cannot use any calc method variables.
- Matching rows flagged as GoTo or Change are ignored.

The idea is that this behavior only makes sense if the multiple insertions can take place without requiring any further input from the user. Any configuration that requires additional input, such as selecting a calc method or completing variable values, is considered invalid and will prevent the insertion.

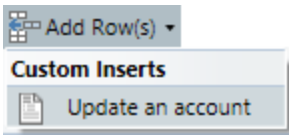
These limitations apply regardless of whether the process finds one match or multiple matches. Once the multiple insert behavior is enabled, the limitations apply in all cases.

When the insertion process is complete, the user's cursor will be placed at the first insertion location. Keep in mind that the user has no way of knowing the other insertion locations except by visual cues in the sheet itself. You should design your templates and calc methods with this in mind.

## The dynamic insertion process

This section explains how the dynamic insertion process works, to help template designers construct the sheet and Excel logic appropriately.

1. If a DynamicCMColumn tag exists in the sheet, the label text for that column displays on the **Add Row(s) > Custom Inserts** menu as a stand-alone item. If any InsertCMColumn tags exist in the file, they still display on the menu as normal. The only difference between the DynamicCMColumn entries and the InsertCMColumn entries is that the DynamicCMColumn entries do not have a sub-menu. For example:

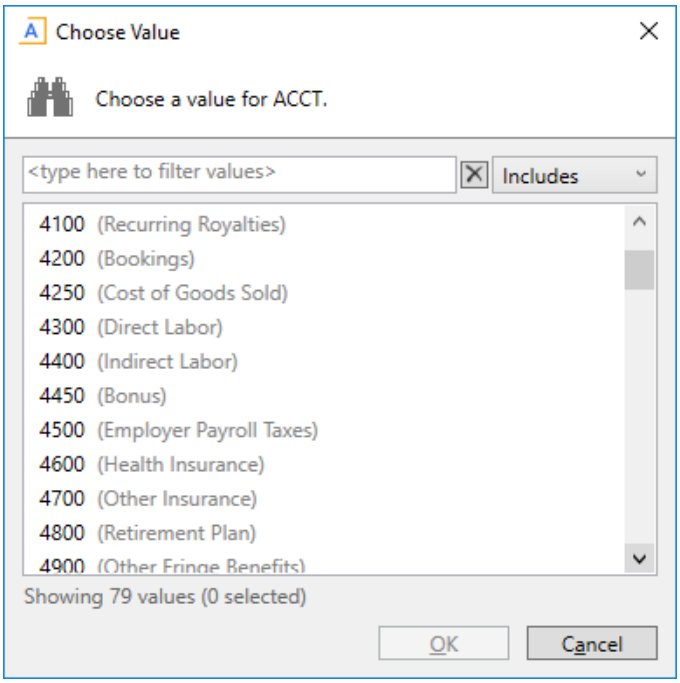



2. The user is prompted to select a single item from the table specified in the DynamicCMColumn tag. The list of items is filtered based on any optional filter placed in the tag.

   For example, if the DynamicCMColumn tag is:

   ```
   [DynamicCMColumn;Update an account;InsertHere; acct.acct|acct.category
   NOT IN ('balance sheet','statistics')|Budget!BB22|F1]
   ```

   Then the user selection dialog will display like this:




3. Once the user selects an item, it is placed in the SheetDestination cell designated in the DynamicCMColumn tag. The sheet is then calculated.

   Continuing the example DynamicCMColumn tag above, the item would be placed in cell BB22.

4. After the calculation, Axiom does one of the following to find the active row:
    - If the DynamicCMColumn tag has a defined DynamicMarker, Axiom looks for a matching marker within an InsertCM tag in that column. The first matching tag in the column becomes the active row.
    - If the DynamicCMColumn tag does not have a defined DynamicMarker, Axiom looks for the first row in the column that contains an InsertCM tag. That row becomes the active row.

    The example DynamicCMColumn tag above uses a marker of InsertHere. Therefore the first InsertCM row that contains the InsertHere marker will be the active row. For example:

    ```
    [InsertCM;;Input Monthly;;Insert;Above;;InsertHere]
    ```

    The way in which a particular row gets identified as the active row is entirely up to the spreadsheet logic developed by the template designer. Typically the InsertCM tags are constructed using a formula, so that once the sheet is calculated only one row is eligible to be the active row, based on some calculation involving the user's selected item.

    **NOTE:** If MultipleInsert is enabled for the DynamicCMColumn tag, then the process will check for all matching rows, instead of stopping at the first matching row in the column.

5. If the InsertCM tag in the active row is configured to allow calc method insertion, then a calc method is inserted as follows:
    - If only one calc method is valid for insertion at that location, that calc method is inserted. If the MaxInsertCount parameter of the InsertCM tag is 0 or above, then the user is first prompted to specify how many instances of the calc method they want to insert, up to the specified limit. If this parameter is blank, then the user will not be prompted and only one instance of the calc method will be inserted.
    - If multiple calc methods are valid for insertion at that location, then the **Insert Calc Method** dialog opens, so that the user can select which calc method to use. This dialog contains an option to specify multiple items to insert if desired (the MaxInsertCount limit specified in the InsertCM tag, if any, will be honored).
    - If the selected calc method uses variables, a calc method form is presented to the user before the calc method is inserted.
    - The calc method is inserted either above or below the active row, as configured in the InsertCM tag. The user's selected item and any variable values are placed within the inserted calc method as applicable (if the DynamicCMColumn tag has a defined CalcMethodDestination, and if the inserted calc method has defined variables). If applicable, action tags are processed after the calc method insertion as normal.

In the InsertCM example above, the row is configured for insert, and Input Monthly is the only allowed calc method. The InsertCM tag is not configured to prompt for multiple lines, and the calc method does not use variables, so the Input Monthly calc method is automatically inserted with no further input required from the user. The calc method is inserted above the current row. The DynamicCMColumn tag has a defined CalcMethodDestination of F1, so the selected item is placed in the first row of the calc method, in column F.

**NOTE:** If MultipleInsert is enabled for the DynamicCMColumn tag, then calc method insertion will only take place if all matching rows meet the requirements as described in the previous section. Essentially, the insertion must be able to occur without requiring any further input from the user.

6. If the InsertCM tag in the active row is configured for GoTo (or Change), then the user is taken to the active row and no further action is taken.

   For example, if the active row was instead configured like this, then the user would simply be taken to the active row:

   `[InsertCM;;Input Monthly;;GoTo;;;InsertHere]`

   **NOTE:** If MultipleInsert is enabled for the DynamicCMColumn tag, then any rows configured for GoTo or Change are ignored.

7. If no active row is found, then the user is informed that the insertion cannot occur. This occurs if:
   - The DynamicCMColumn has a defined dynamic marker, but no InsertCM tags in the column contain the marker.
   - The DynamicCMColumn does not have a defined dynamic marker, and no InsertCM tags were found in the column.

## Calc method control tag syntax

This section provides a detailed explanation of the calc method control tag syntax, and several examples.

### Standard insertion control syntax (InsertCMColumn)

The control column syntax for standard calc method controls is as follows:

`[InsertCMColumn; HeaderLabel]`

The InsertCMColumn portion of the tag is required, and flags the column as the control column.

*HeaderLabel* is the text to appear on the **Add Row(s) > Custom Inserts** menu, for this set of enabled rows. The header text could be the name of the sheet, or it could describe the allowed actions. For example:

- `[InsertCMColumn; Add Rows to Budget Sheet]`

- `[InsertCMColumn; Add New Employees]`

The header label parameter is optional and may be omitted if not necessary. For example, if you are enabling a calc method control column for the sole purpose of enabling rows for overwriting, then no insertion points will display on the menu anyway, and you can omit the header text.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

## Dynamic insertion control syntax (DynamicCMColumn)

The control column syntax for dynamic insertion controls is as follows:

`[DynamicCMColumn; Label; DynamicMarker; Table|Filter|SheetDestination|CalcMethodDestination; MultipleInsert]`

The DynamicCMColumn portion of the tag is required, and flags the column as the control column. The remaining parameters must be completed as follows:

| Item | Description |
|---|---|
| Label | The text to appear on the **Add Row(s) > Custom Inserts** menu. For example, "Update an account".<br><br>This is the only text that will appear on the menu for this column. Unlike when using the InsertCMColumn tag, any label text defined in the individual InsertCM tags is ignored. |
| DynamicMarker | Optional. The text that identifies an InsertCM row as the "active" row for dynamic insertion.<br><br>For example, you could use the text "InsertHere" as the dynamic marker. After the user selects an account and the sheet is calculated, Axiom will look for the first InsertCM tag in the column that contains a matching "InsertHere" tag. That row becomes the active row for insertion.<br><br>If omitted, then the active row is the first row in the column that contains an InsertCM tag. |

| Item | Description |
| --- | --- |
| Table | The table to present to the user, so that the user can select the desired item in the table. The table can be any reference table, such as ACCT or DEPT.<br><br>The full syntax for specifying the table is as follows:<br>`Table.PrimaryColumn.AdditionalColumn.AdditionalColumn`<br><br>The PrimaryColumn is the column that contains the items for the user to select. In most cases this is the key column for the table.<br><br>Any additional column listed will display in the dialog to help the user make a selection. You can list as many additional columns as desired, delimited by periods.<br><br>For example:<br>`ACCT.ACCT.Description`<br><br>In this case the user will select an account from the ACCT table, and the Description column will also display in the dialog.<br><br>**NOTE:** Additional columns only apply if the PrimaryColumn is the key column for the table. Otherwise they will be ignored. |
| Filter | Optional. A filter to limit the items to display in the selection dialog.<br><br>The filter must be based on the specified table. Use standard filter criteria syntax. For example:<br>`ACCT.Category NOT IN ('Balance Sheet','Statistics')`<br><br>Accounts belonging to these categories will not be listed in the dialog.<br><br>If omitted, then all items in the table will be listed in the dialog. If you omit this parameter, you must delimit it with an empty pipe character. |
| SheetDestination | The cell in the workbook to place the user's selection. Specify both the sheet name and the cell reference, even if you want the item to be placed on the current sheet (the sheet where the tag is defined). For example: Budget!B25.<br><br>This is a fixed location. The spreadsheet logic that you build within the sheet to determine the active row for insertion must point to this cell to obtain the user's selected item. |

| Item | Description |
|---|---|
| CalcMethod Destination | Optional. The cell within the inserted calc method to place the user's selection. This location is relative to the calc method.<br><br>The location is specified as follows: `<ColumnLetter><CalcMethodRow>`.<br><br>For example:<br>• If the value should be placed in column B of the calc method, within the first row of the calc method, enter the following: `B1`.<br>• If the value should be placed in column B of the calc method, but within the third row of the calc method, enter the following: `B3`.<br><br>If omitted, then the user's selection is not placed inside the inserted calc method.<br><br>**NOTES:**<br>• This location must apply to all calc methods that could potentially be inserted by use of the dynamic insertion controls.<br>• This item is ignored if the action performed on the active row is simply to navigate ("goto") an existing row rather than insert a new calc method. |
| MultipleInsert | Optional. Type the keyword `MultipleInsert` if you want the dynamic insertion to allow multiple insertion points. Omit this parameter to use the standard behavior of a single insertion.<br><br>If enabled, then a calc method will be inserted at all matching locations, not just the first matching location. Enabling this behavior has the following limitations:<br>• All matching rows must be limited to a single calc method. This means that the InsertCM tag for the matching row must either specify a single allowed calc method, or a calc method group that only contains one calc method. Also, the MaxInsertCount parameter for the InsertCM tag must be blank (meaning inserting multiple instances of the calc method is not allowed).<br>• The allowed calc method for the matching row cannot use any calc method variables.<br>• Matching rows flagged as GoTo or Change rows are ignored.<br><br>The idea is that multiple calc method insertions can take place without requiring any further input from the user. Any configuration that requires additional input, such as selecting a calc method or completing variable values, is considered invalid and will prevent the insertion. |

All required parameters must be present in the tag in order for the label text to appear on the **Add Row (s)** menu.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

### Examples of DynamicCMColumn tags

```
[DynamicCMColumn; Update an account;InsertHere;
acct.acct.description.category|acct.category NOT IN ('balance
sheet','statistics')|Budget!BB22|F1]
```

This example places the text "Update an account" on the **Add Row(s) > Custom Inserts** menu. When a user clicks on this item, a dialog opens so that the user can select an account. In addition to the list of accounts, the Description column and the Category column also display in this dialog. The list of accounts is filtered to not include accounts that belong to the Balance Sheet and Statistics categories.

After the user selects an account, the selected item is placed into cell BB22 of the Budget sheet, and the sheet is calculated. The active row for dynamic insertion is then identified by looking for the first row in the column that contains an InsertCM tag with the matching dynamic marker text of "InsertHere". If a calc method is inserted as a result of the dynamic insertion process, then the selected account is placed in the first row of the inserted calc method, in column F.

```
[DynamicCMColumn; Update an account; ; acct.acct||Budget!BB22]
```

This example is the same as the prior example, except that only the required parameters have been specified. This is provided to contrast the difference in behavior when certain optional parameters are omitted. In this case:

- No dynamic marker is used, so the active row will be determined by locating the first row in the column that contains an InsertCM tag.
- No additional columns are specified for the table, so the selection dialog will use the "simple view" (account codes with the description column appended in parentheses).
- No filter is specified for the table, so all accounts will be listed in the selection dialog.
- No calc method destination is specified, so the selected item will not be placed in any inserted calc method.

```
[DynamicCMColumn; Add a new account; InsertHere;
acct.acct.description.category| |Budget!BB22|F1;MultipleInsert]
```

This example uses the MultipleInsert parameter to allow calc method insertion at multiple locations, not just the first matching location. The user would select an account and that account might be inserted at one or more locations in the sheet (in this example, the plan file might contain several departments so multiple instances of the same account would make sense).

### Enabled row syntax (InsertCM)

The enabled row syntax for calc method controls (standard and dynamic) is as follows:

`[InsertCM; InsertionPointLabel; CalcMethodList; GroupList; AllowedAction; InsertLocation; MaxInsertCount; DynamicMarker]`

The InsertCM portion of the tag is required, and marks the row as enabled for a calc method action. The allowed actions, allowed calc methods, and navigation depend on the optional parameters:

| Parameter | Description |
|---|---|
| InsertionPointLabel | Defines the text to display on the **Add Row(s)** menu of custom insertion points, organized under the header text of the control column.<br><br>**NOTE:** This parameter only applies when using standard calc method controls (InsertCMColumn). It is ignored when using dynamic insertion controls (DynamicCMColumn), but must still be delimited with an empty semicolon.<br><br>This parameter only applies to rows that are enabled for insert (either insert only, or insert and overwrite). When an insertion point is selected from the menu, the action is always to insert a calc method at that location.<br><br>If omitted, then this insertion point does not display on the **Add Row(s)** menu, but users can still insert calc methods manually at this location (if the row is enabled for insert).<br><br>Custom insertion points cannot be used to navigate users to a location for overwriting. If the only action allowed on the row is overwriting, then this parameter is ignored. |
| CalcMethodList | Defines the list of allowed calc methods for the row. Separate multiple calc method names with commas.<br><br>If a calc method list is defined, then only calc methods that match this list can be used to insert or overwrite at this location (unless a group list is also defined).<br><br>If both the CalcMethodList and GroupList are omitted, then any calc method can be used to insert or overwrite at this location. |
| GroupList | Defines the list of allowed calc method groups for the row. Separate multiple group names with commas.<br><br>If a group list is defined, then only calc methods that belong to the specified groups can be used to insert or overwrite at this location (unless a calc method list is also defined). |

| Parameter | Description |
| --- | --- |
| AllowedAction | Specifies the allowed action for this location. The action can be one of the following:<br>• `Insert`: Users can insert calc methods at this location.<br>When using dynamic insertion, a calc method will be automatically inserted at this location if the row is identified as the active row.<br>• `Change`: Users can change calc methods at this location. The keyword "Overwrite" is also recognized here, for backward-compatibility.<br>When using dynamic insertion, the user will be taken to this location if the row is identified as the active row, but no further action occurs. If the user has the rights to change calc methods, they can use the **Add Rows > Change Calc Method** option to do so.<br>• `Goto`: When using dynamic insertion, the user will be taken to this location if the row is identified as the active row. Insert and/or overwrite are not allowed at this location.<br>**NOTE:** This option only applies when using a dynamic insertion control column (DynamicCMColumn). If you are using a standard calc method control column (InsertCMColumn) and GoTo is specified as the allowed action, then this row does *not* display on the custom inserts menu, and no calc method actions can be performed on it.<br>• <Blank>: Users can insert and change calc methods at this location.<br>When using dynamic insertion, a calc method will be automatically inserted at this location if the row is identified as the active row. |
| InsertLocation | Specifies where a calc method will be inserted relative to the current row, if the row is enabled for insertion:<br>• `Above` (or blank): New calc methods are inserted above the current row.<br>• `Below`: New calc methods are inserted below the current row. |

| Parameter | Description |
| --- | --- |
| MaxInsertCount | Specifies the maximum number of calc method instances that can be inserted at this location, if the row is enabled for insertion. For example, if the user is inserting detail rows, you may want to give the user the option to insert multiple detail rows at a time.<br><br>The behavior of this option depends on how many calc methods are available for insert at this location:<br><br>• If the **Insert Calc Method** dialog will display to the user (i.e. more than one calc method is available for insert at this location), then this setting controls what the user can enter into the **Number of items to insert** field in the dialog. In this circumstance, blank or 0 means unlimited—the user can insert as many instances as they wish. You can enter a number if you wish to limit the number of instances that can be inserted, or you can enter 1 if you do not want the user to be able to insert more than one instance of the calc method.<br>• If only one calc method is available for insert at this location, then this setting controls whether or not the user will be prompted to insert more than one instance of the calc method, and if so what is the maximum number they can input. You should leave this setting blank if you do not want the user to be prompted. If you want the user to be prompted, you can enter 0 for unlimited instances, or enter a number higher than 1. (Entering 1 does not make sense in this circumstance, because this will cause the prompt to display but the user will be unable to change it to a number higher than 1.) |
| DynamicMarker | This item is only applicable when used in a dynamic insertion control column that has a dynamic marker defined in the DynamicCMColumn tag.<br><br>After a user selects the desired account (or other dimension) and the sheet is calculated, the "active" row for insertion is determined as the first row in the column with a matching dynamic marker.<br><br>Use of a dynamic marker is optional. If no dynamic marker is specified in the DynamicCMColumn tag, the active row is determined as the first row in the column with an InsertCM tag. |

The CalcMethodList and GroupList are combined together to result in the full list of calc methods allowed at the location. You can use one or the other, or both. Note the following:

- If only one calc method is allowed, that calc method is automatically inserted or used to overwrite when the user performs the calc method action on the row. The calc method dialog does not display to the user.
- If multiple calc methods are allowed, then the calc method dialog is filtered to only show the allowed calc methods.

If a parameter is omitted, and you want to specify a parameter after that parameter, you must delimit the omitted parameter with a semicolon.

**NOTES:**

- If you want calc method actions to be available on inserted calc method rows within a sheet (whether inserted via Axiom query or by a user), then the calc method itself must be designed so that the InsertCM tag is saved as part of the calc method.
- If the calc method is a multiple-row calc method, and you want to allow users to change it, it is not necessary to tag all rows with an InsertCM tag. Typically, only the first row is marked with an InsertCM tag and enabled for change calc method, but any tagged row within the calc method will trigger the ability to change the calc method.
- The InsertCM tag can be placed within a formula if desired. If you are using dynamic insertion, it is essentially a requirement (so that the tags can adjust dynamically based on the user's dimension selection). In this case, you may want to use the right-click helper dialog to construct the initial tag, and then modify the tag to convert it to a formula. Once the tag is within a formula, you will be unable to make further modifications using the right-click dialog.

### Examples for use with a standard calc method control column (InsertCMColumn)

```
[InsertCM;]
```

In this example, users can manually insert or overwrite using any calc method at this location. Because no insertion point label is defined, no text appears on the **Add Row(s)** menu for this location. Note that if you omit all other parameters, you must include the first delimiting semicolon.

```
[InsertCM; Insert New Employee]
```

In this example, users can manually insert or overwrite using any calc method at this location. Users can also insert at this location by selecting the "Insert New Employee" insertion point from the **Add Row(s)** menu.

```
[InsertCM; Insert New Employee; Add New Salaried Employee, Add New Hourly
Employee; ; Insert]
```

In this example, users can insert one of the two specified calc methods at this location. The GroupList parameter is not used, and is delimited with an "empty" semicolon. Users can only insert calc methods at this location, as specified by the AllowedAction parameter. Overwriting is not allowed.

```
[InsertCM; Add Employees; ; Payroll]
```

In this example, users can insert or overwrite using any calc method that belongs to the Payroll group. The CalcMethodList parameter is not used, and is delimited with an "empty" semicolon.

```
[InsertCM; ; Input Monthly; Budget ;Change]
```

In this example, users can change calc methods using the Input Monthly calc method, or using any calc method that belongs to the Budget group. Because users cannot insert at this location, the InsertionPointLabel parameter is not used, and is delimited with an "empty" semicolon. In order to

change the calc method, a user must manually place the cursor at this location and select **Add Row(s) > Change Calc Method**.

```
[InsertCM; Insert New Employee; New Employee; ; Insert; Above; 5]
```

In this example, only one calc method, New Employee, is allowed for insertion at this location. When the user selects this insertion point, the specified calc method is automatically inserted at the location. The **Insert Calc Method** dialog does not display (even if the user is an administrator). However, this example also uses the MaxInsertCount parameter, so the user will first be prompted to specify how many instances of the calc method to insert (up to 5).

### Examples for use with a dynamic insertion control column (DynamicCMColumn)

In the following examples, the current row is being identified as the active insertion row by the presence of the DynamicMarker text "InsertHere" in the final parameter. This assumes that the DynamicCMColumn contains the same DynamicMarker text. If the DynamicCMColumn does not use a dynamic marker, then the described actions would only apply if the row is identified as the active row by being the first row marked with InsertCM in the column.

```
[InsertCM; ; Budget Detail; ; Insert; Below; 0; InsertHere]
```

In this example, the row is enabled for insert, and only the Budget Detail calc method is eligible for insert, so the user will not be prompted to select a calc method. The MaxInsertCount parameter is used so that the user will be prompted to specify how many items to insert; since the number is 0, the user can insert an unlimited number of detail rows. Finally, since the InsertLocation is specified as below, the rows will be inserted below the current row (instead of the default behavior of above).

```
[InsertCM; ; ; ; GoTo; ; ; InsertHere]
```

In this example, the AllowedAction parameter is set to GoTo, so the user is taken to the current row and no calc method is inserted. Since calc method insertion is not possible on this row, the CalcMethodList, GroupList, InsertLocation, and MaxInsertCount parameters are irrelevant and therefore left blank. This type of tag is intended for use on existing rows, so that if a user selects a dimension item that already exists in the sheet, they are simply taken to the existing row.

```
[InsertCM; ; ; BudgetGroup; Change; ; ; InsertHere]
```

This example is similar to the previous example, except in this case the AllowedAction is Change instead of GoTo. The dynamic insertion behavior is the same: the user is taken to the current row and no calc method is inserted. However, if the user has **Allow Calc Method Change** permissions, they have the additional option of changing the existing row with another calc method (in this case, any calc method that belongs to the BudgetGroup). The user must explicitly go to the **Add Row(s)** menu if they want to perform the change; the dynamic insertion process does *not* open the **Change Calc Method** dialog or prompt the user in any way.

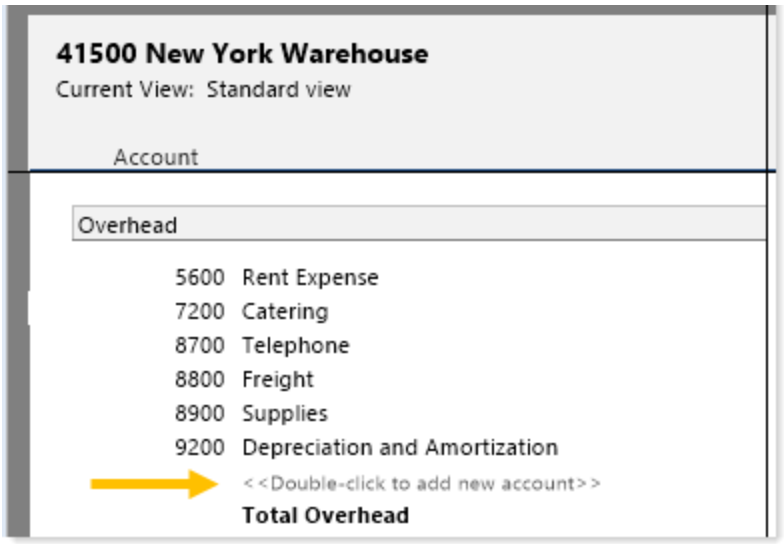
# Setting up double-click insertion for calc methods

You can set up your templates (or other files that support use of calc method libraries) so that end users in plan files can insert calc methods by double-clicking on a row, instead of needing to use the **Add Row(s)** menu item. To do this, you use the GetCalcMethod function in the row where you want user to insert calc methods.

GetCalcMethod is a simple function that takes a single parameter:

```
GetCalcMethod("DisplayText")
```

The DisplayText is the text to display in the cell, such as "Double-click to add a new account". By default, the text displays as normal text in the cell. If you want the text to look like a clickable hyperlink, you must manually format the cell text (for example, blue and underlined).




*Example use of GetCalcMethod*

**NOTES:**

- The placement of the function in the sheet depends on whether you are using the optional parameter of the InsertCM tag to control the insertion location. By default, calc methods are inserted *above* the current row. However, you can configure the InsertCM tag to specify that calc methods are inserted below the current row if desired.
- Make sure that the column which contains the GetCalcMethod function is wide enough to contain all of the display text. If the display text is wider than the column and spills into adjacent columns, then the user may end up double-clicking in the wrong place. For example, if you place the function in column F, and then make column F very small so that most of the text displays in column G, users may end up clicking on column G and nothing will happen.

### GetCalcMethod Requirements

When a cell containing the GetCalcMethod function is double-clicked, a calc method will be inserted at the current location, if the following conditions are met:

- The user has the appropriate file group permissions to insert calc methods.
- The row is enabled for calc method insertion (via standard calc method controls), or calc method controls are not being used in the sheet.

If either of the above conditions are not met, then an appropriate error message is displayed to the user.

**NOTE:** If dynamic calc method controls are used in the sheet (DynamicCMColumn), then GetCalcMethod cannot be used to insert calc methods. The function will behave as if no rows are enabled for insertion. However, if the sheet uses *both* dynamic controls and standard controls, then the function will work as expected with enabled rows in the InsertCMColumn.

### GetCalcMethod Behavior

If no calc method controls are used in the sheet, then double-clicking GetCalcMethod behaves in the same way as using **File Options > Add Row(s) > Insert Calc Method(s)**. The user selects from any calc method in the library, and that calc method is inserted.

If calc method controls are used in the sheet, then the presence or absence of an InsertCM tag on the row controls the behavior when GetCalcMethod is double-clicked:

- If the row does not have an InsertCM tag, or if an InsertCM tag is present but it does not allow insertion, then no calc method insertion is performed. An appropriate error message is displayed to the user.
- If the row has an InsertCM tag that allows insertion, the settings in the tag control the insertion. Settings such as allowed calc methods, insert location, and prompting for insert count are all honored. The behavior is the same as if the user had selected this row as a custom insertion point. If only one calc method is allowed, it is automatically inserted (including the optional prompt for inserting multiple items, if applicable). If multiple calc methods are allowed, the user selects one of the calc methods.

# Using calc method libraries with Axiom queries

When you are creating an Axiom query for use in a plan file (or any other file that supports use of calc method libraries), you can use the sheet's calc method library to apply formatting and formulas to the data rows, instead of using an in-sheet calc method. This approach is typically used to build out the planning rows of plan files. Users can then complete data inputs, add rows as necessary, and even change the calc method used on a row if desired.

To use an Axiom query in a plan file, you first create it in the template. Typically the query would be configured so that when plan files are built from templates, Axiom queries data for the applicable plan code, applies calc methods from the library, and populates the file with data. Usually the template itself does not contain many "fixed" rows of data, so the Axiom queries are used to dynamically insert the accounts or employees relative to the particular plan code for each plan file.

When using a calc method library, each record of data can be assigned a different calc method. For example, some accounts might use an "Input Monthly" calc method for budgeting, while others might use a "Spread Total" calc method. Different formatting and calculations are applied to different accounts as needed.

Each Axiom query in a template can use different calc method settings. For example, you could have one Axiom query that uses an in-sheet calc method to populate the top section of the sheet, and another Axiom query that uses the calc method library to populate the bottom section of the sheet.

The following requirements apply to using calc method libraries with Axiom queries:

- Calc method libraries only apply to templates and plan files. Report files and driver files do not have access to a calc method library, and therefore must use an in-sheet calc method for Axiom queries.
- Calc method libraries can only be used with standard, vertical Axiom queries. Calc method libraries are row-based, and therefore cannot be used with horizontal Axiom queries. If you want to use a horizontal Axiom query in a template or a plan file, then you must use an in-sheet calc method with that query.

**IMPORTANT:** In order to use a calc method library, the in-sheet calc method setting for the Axiom query must be blank. If an in-sheet calc method is specified for the Axiom query, then that setting will take priority over any other calc method settings for the query.

## Calc method assignment options

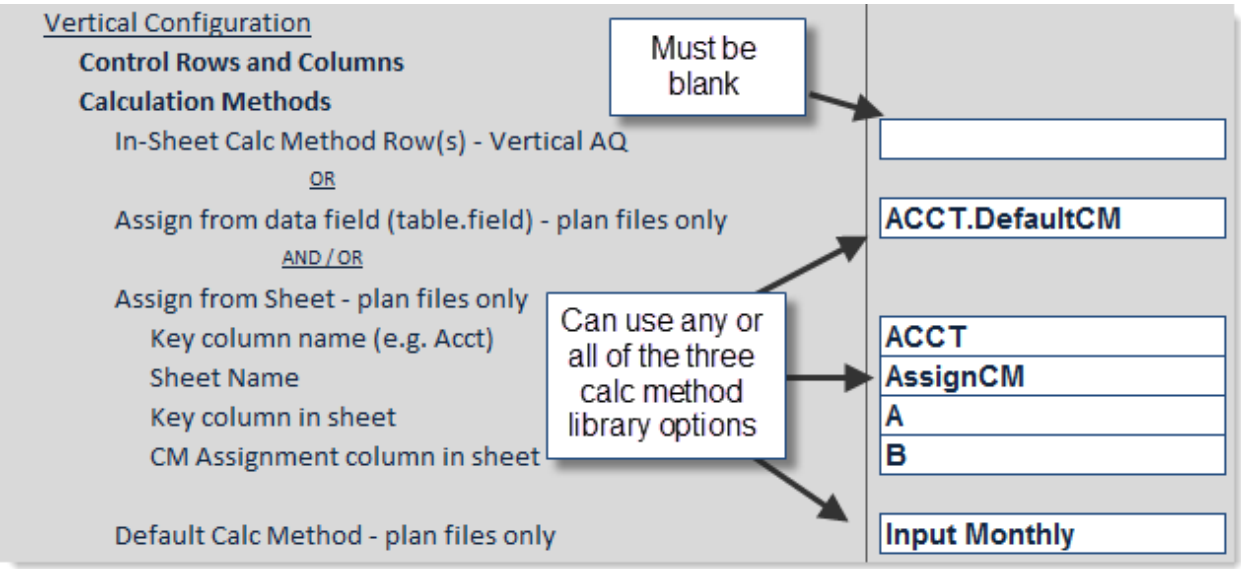
In order to use a calc method library with an Axiom query, Axiom must have a way of determining which calc method to apply to each record in the query. There are several ways to configure this lookup.

- **Look up assignments from a database table**: You can configure the Axiom query to look up calc method assignments from a table in the database. There are two ways to do this:
    - **Single assignment column**: The simplest and most common approach is to set up a single assignment column in the appropriate database table (for example, the ACCT table), and then point the Axiom query to that column to look up the assignments.
    - **Multiple assignment columns by plan code groupings**: If calc method assignments need to vary by a plan code grouping, such as the type of department, then you can use multiple assignment columns. In this case, you create multiple assignment columns in the appropriate database table (such as the ACCT table). Then in the plan code table, you create a "master" assignment column that maps each plan code to its appropriate assignment column in the ACCT table.

- **Read assignments from a sheet**: You can configure the Axiom query to read calc method assignments from a sheet in the file, instead of (or in addition to) looking them up from a database column. This approach requires the most manual setup, but it is the most flexible and can be used for virtually any circumstance.
- **Default calc method**: You can use the default calc method on its own, or in conjunction with the other options. If a calc method's assignment is left blank using any of the options described above, then the default calc method will be used for that record. If desired, you can specify *only* the default calc method (leaving all other calc method options blank), and then the default calc method will be used for all records in the query.

Multiple calc method assignment options can be used in the same query. For example, you can use the sheet option, the database table option, and the default calc method option together. For each record of data in the query, the query first checks the sheet, then checks the database column, and lastly uses the default calc method.

All calc method library assignment options are specified on the Control Sheet for the Axiom query. You must edit the Control Sheet directly in order to use the calc method library, because calc method library settings do not display in the Sheet Assistant. The options are located in the **Vertical Configuration** section of the Axiom query, in the **Calculation Methods** subsection. To use any of these options, the **In-Sheet Calc Method Row(s)** setting must be left blank.



**NOTES:**

- If a calc method assignment is invalid, then an error results and the Axiom query stops processing. For example, if you specify "InputMonthly" as the calc method assignment, but the actual name of the calc method in the library is "Input Monthly," then no match is found and the assignment is invalid. The default calc method is not used in this case.
- When drilling down an Axiom query that uses the calc method library, the default calc method is used for the drill results, regardless of whichever calc method was used on the row being drilled. If no default calc method is specified for the query, then no calc method is applied to the drill results. This means that the drill results will only present the raw data resulting from the field definition—no formatting and formulas from the row will be carried over to the drill sheet.

### Using a single assignment column

To use a single calc method assignment column for an Axiom query:

1. In the appropriate database table, create a column to contain the calc method assignments for each item in the table.

   For example, if the Axiom query is being populated with accounts, you would create a column in the ACCT table and then enter valid calc method names into that column. The column can be named anything you like (for example, AssignCM).

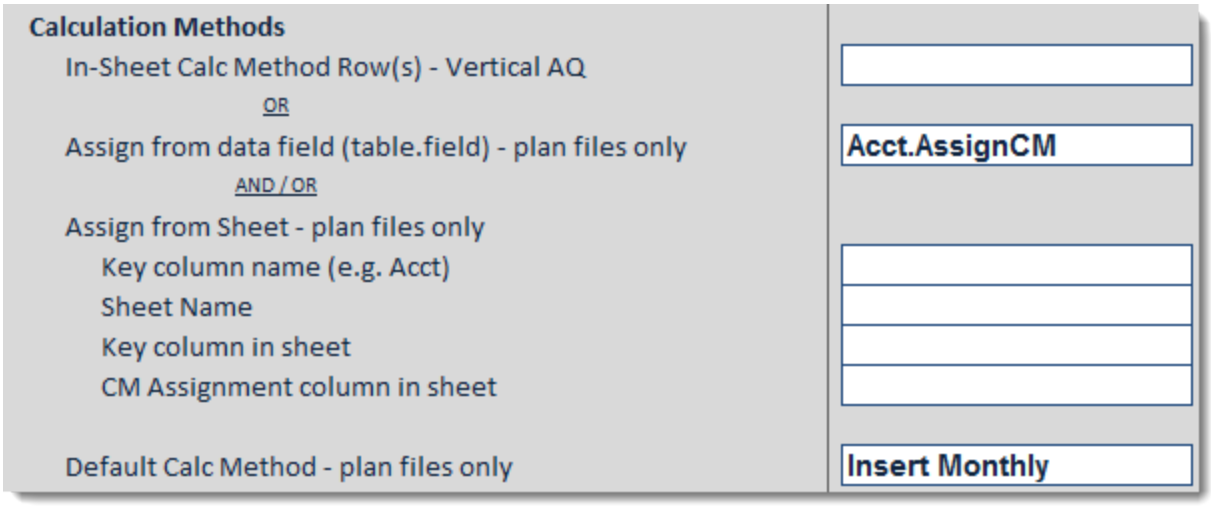
   You can use the same assignment column for multiple Axiom queries if appropriate.

   **NOTE:** If all records in the table need to use the same calc method for this Axiom query, then you do not need to create an assignment column; you can use the default calc method setting instead.

2. Complete the following settings in the Control Sheet for the Axiom query. These settings are located in the **Vertical Configuration** section of the query, under the heading **Calculation Methods**.

| Item | Description |
|---|---|
| Assign from data field | Enter the name of the column that contains the relevant calc method assignments for this Axiom query, using fully qualified Table.Column syntax. For example, `ACCT.AssignCM`. When data records are inserted into a data range, Axiom looks to this column to determine which calc method to use for each record. |
| Default Calc Method | Optional. Enter the name of a calc method to be used if an entry in the assignment column is blank. If an entry in the assignment column is invalid, then the default calc method is *not* used; an error results instead. |

The following screenshot shows an example configuration using a single calc method assignment column:




## Using multiple assignment columns by plan code type

The basic implementation of the calc method assignment column assumes that all plan codes (for example, departments) use the same calc method for any single account. If necessary, you can configure the Axiom query to use different assignment columns depending on the type of plan code.

For example, most plan codes might plan for supplies by inputting a total annual value that is spread evenly over months. But for other plan codes, it might be more appropriate to calculate supplies based on a monthly statistic. You can configure the Axiom query so that all plan codes in the first category use a Spread Total calc method for the supplies account, whereas the plan codes in the second category use a Monthly Statistic calc method.

**To use different calc method assignments by plan code:**

1. Create multiple assignment columns in the appropriate table (for example, ACCT), and complete the columns with the calc method assignments appropriate for a particular type of plan code. The columns can be named anything you like.

   For example, you might have one column that contains the assignments for revenue departments (CMRevenue), and one column that contains the assignments for overhead departments (CMOverhead).

2. In the plan code table (for example, DEPT), create a "master" assignment column. The column can be named anything you like (for example, MasterAssignCM). In this column, for each department, enter the name of the appropriate assignment column from the ACCT table.

   For example, if the department is a revenue department, enter CMRevenue into the MasterAssignCM column.

Every code in the plan code table must have an entry in the master assignment column (unless you are not creating a plan file for that code, or you know that the plan file for that code does not use this Axiom query).

3. In the Control Sheet settings for the Axiom query, in the **Assign from data field** box, enter the column name for the master assignment column, using fully qualified Table.Column syntax. Then, in parentheses, enter the name of the table containing the detailed assignment columns. The syntax is as follows:

   ```
   MasterTable.MasterAssignCM(DetailTable)
   ```

   For example, if the master column is in the DEPT table and is named MasterAssignCM, and the detailed columns are in the ACCT table, then you would enter the following:

   ```
   DEPT.MasterAssignCM(ACCT)
   ```

   When data records are to be inserted into a data range, Axiom first looks to the MasterAssignCM column in the DEPT table, and finds the entry for the current department. Axiom then finds the specified column in the ACCT table and uses that column to assign calc methods to accounts.

You can optionally complete the **Default Calc Method** setting. If an entry in the assignment column is blank for a particular account, the default calc method will be used. If an entry in the assignment column is invalid, then the default calc method is *not* used; an error results instead.

### Reading calc method assignments from a sheet

Using this option, you set up a sheet in the file to contain the calc method assignments, and then you configure the Axiom query to read the assignments from the sheet.

You can use any methodology to create the list of assignments in the sheet. You can manually type in a list of assignments, or you can use an Axiom query to bring in a list that is stored somewhere in the database. If you set up the Axiom query to be dynamic based on criteria such as the current plan code, you can retrieve a different list of assignments for each plan code. This is more flexible than pointing the Axiom query at the database directly using the **Assign from data field** option, because you can store the assignments anywhere in the database, and use any criteria to retrieve them.

**To read calc method assignments from a sheet:**

1. Set up a sheet in the file to contain the calc method assignments. Within the sheet, one column must contain the relevant key codes (such as ACCT), and one column must contain valid calc method names.

   The sheet can be hidden or visible. Within the target columns, only rows that contain valid key codes will be evaluated for calc method assignments; content such as Axiom query field definitions and column headers will be ignored.

**IMPORTANT:** If you are using an Axiom query to retrieve the list of assignments, you must make sure that query runs before the query that will use the assignments. For example, you might set the query to **Refresh on Open**. If the query does not need to refresh on open, or if both queries need to refresh on open, then make sure that the sheet for the assignment query is located before (to the left of) the sheet for the query using the assignments (on the Control Sheet, not the literal sheet order). If both queries are on the same sheet, then the assignment query must have a higher AQ number than the query using the assignments.

2. Complete the following **Assign from sheet** settings in the Control Sheet for the Axiom query. These settings are located in the **Vertical Configuration** section of the query, under the heading **Calculation Methods**.

| Item | Description |
|---|---|
| Key column name | The name of the key column for the calc method assignments.<br><br>For example, if data is being brought into the sheet by accounts, then enter ACCT. |
| Sheet Name | The name of the sheet that holds the calc method assignments.<br><br>For example, you may have created a sheet named CMAssign to hold the assignments. |
| Key column in sheet | The column in the sheet that contains the key codes.<br><br>For example, if the key column is ACCT, and the account codes are located in column C of the CMAssign sheet, then you would enter C. |
| CM Assignment column in sheet | The column in the sheet that contains the calc method assignments.<br><br>For example, if assignments are in column D of the CMAssign sheet, you would enter D. |

**NOTE:** If you do not see these settings in your Control Sheet, then you must update the Control Sheet. You can do this using the Sheet Assistant.

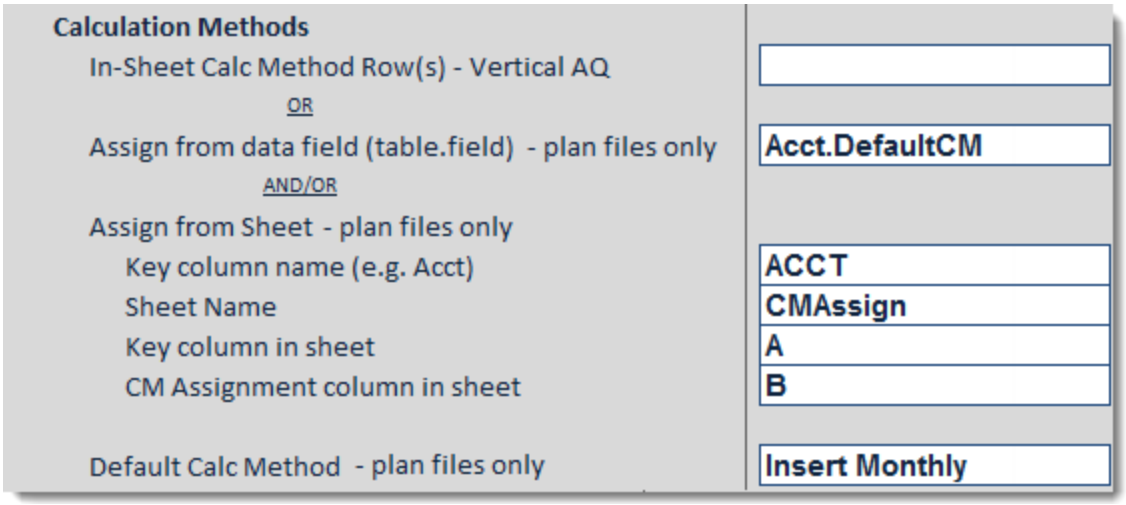
All four settings are required in order to read assignments from the sheet. If you specify the key column name but leave any other settings blank, an error will result when the query is run. If you specify any other option but leave the key column blank, then all settings in this section are ignored.

You can also complete the following optional settings:

- **Default Calc Method**: If an entry in the sheet is blank for a particular account, or if a particular account is not listed in the sheet at all, then the default calc method will be used for that account. If an entry in the sheet is invalid, then the default calc method is *not* used; an error results instead.

- **Assign from data field**: If both the **Assign from sheet** and **Assign from data field** options are completed, then Axiom will look up the calc method assignment for each account as follows:
    - Axiom first checks the specified sheet for the account. If a match is found, that assignment is used.
    - If no match is found on the sheet, Axiom checks the specified table column and uses the assignment there. If the assignment is blank in that column, the default calc method is used.

The following screenshot shows an example configuration using the assign from sheet option:




# Calc method properties

The following properties can be defined for calc methods.

## General properties

The following basic properties apply to all calc methods.

| Item | Description |
|---|---|
| Name | The name of the calc method.<br><br>Users select from these names when inserting or overwriting calc methods. The name should be relatively brief, but descriptive.<br><br>If you change the name of an existing calc method, this may impact settings such as calc method assignments in the plan code table, the specified default calc method for an Axiom query, or calc method controls. You may need to edit these settings for the new calc method name. |

| Item | Description |
|---|---|
| Description | The description of the calc method. Optional.<br><br>This can be used to describe the purpose of the calc method and explain when it should be used. Users can see the descriptions when they select calc methods, so it is a good idea to define a description. |
| Group | The group that the calc method belongs to. Optional.<br><br>Calc method groups can be used in calc method controls, to specify a group of calc methods that are allowed to be used for insertion or overwriting at a particular location. If a group is specified, then any calc method that belongs to the group can be inserted or used to overwrite at that location.<br><br>To define a new group, type the group name into the **Group** field. To assign the calc method to an existing group, select the group name from the drop-down list. When assigning a calc method to an existing group, it is best to select the group name instead of typing it, to eliminate the possibility of typing errors.<br><br>If you change the group for an existing calc method, this may impact where the calc method can be inserted or used to overwrite, if the group name is being used for calc method controls. |
| Variables Use Data Source | Specifies whether calc method variables are defined using a CalcMethodVariables data source in the spreadsheet.<br><br>If enabled, then when the calc method is inserted, Axiom checks for the presence of a CalcMethodVariables data source with a matching sheet name and calc method name. For example, if the calc method library is for sheet Budget and the calc method is named Insert Only, then Axiom looks for the following tag: `[CalcMethodVariables;Budget;Insert Only]`. If a matching data source is found, then it is used to present calc method variables to the user. If it is not found, then the calc method is inserted without variables. For more information, see Defining calc method variables using a CalcMethodVariables data source.<br><br>If disabled, then calc method variables can be defined within the calc method properties, using the **Calc Method Variables** section at the bottom of the dialog. |

## Calc Method Variable properties

This section only applies if **Variables Use Data Source** is not enabled. This section is used to define legacy calc method variables. For more information, see Defining calc method variables within the calc method properties (legacy approach).

Use the icons at the top right of the grid to add, delete, or change the order of variables.

| Item | Description |
| --- | --- |
| Name | The name of the variable. This name displays in the **Calc Method Variables** dialog when users are prompted to select variable values. The name should be brief and descriptive.<br><br>**NOTE:** If the variable type is Related Column Value or GUID, then the variable does not display to the user. In this case the name is for descriptive purposes, to indicate the purpose of the variable to other calc method administrators. |
| Type | The data type of the variable. Select from one of the following:<br><br>• **String**: The user can enter any string value.<br>• **Integer**: The user can enter any whole number.<br>• **Decimal**: The user can enter any numeric value, including decimals.<br>• **List**: The user can select any item in a defined list. List values are presented to the user in a drop-down list.<br>• **Grid**: The user can select any item from a specified table column (for example, ACCT.ACCT to select from a list of accounts). Column values are presented in the **Choose Value** dialog, using a searchable grid.<br>• **Related Column Value**: This variable type is not presented to the user. It is used to return a related column value for a parent Grid variable, so that the value can be referenced within the calc method. The Related Column Value variable must be placed underneath its parent Grid variable.<br>• **GUID**: This variable type is not presented to the user. It generates a globally unique identifier (GUID) and places it into the designated location when the calc method is inserted.<br><br>If the type is **List**, click the **[...]** button in the Type field to define the list. |
| Column Name | The name of the associated column for the variable, using Table.Column syntax. This setting only applies to Grid variables and Related Column Value variables.<br><br>Click the **[...]** button to select a column.<br><br>• For Grid variables, only columns from reference tables or document reference tables can be selected.<br>• For Related Column Value variables, the column must be from the same table as the parent variable. The Column Chooser dialog is automatically filtered based on the table specified for the parent variable. |

| Item | Description |
|---|---|
| Relative Location | The location to place the variable value when the calc method is inserted, relative within the calc method.<br><br>The location is specified as follows: `<ColumnLetter><CalcMethodRow>`.<br><br>For example:<br>• If the value should be placed in column B of the calc method, within the first row of the calc method, enter the following: `B1`. If the calc method is a single-row calc method, the calc method row is always 1.<br>• If the value should be placed in column B of the calc method, but within the third row of the calc method, enter the following: `B3`.<br><br>The entry for the calc method row must be valid within the context of the current calc method. For example, if the current calc method is a three-row calc method, then the only valid entries for the row are 1, 2, and 3. If you attempt to specify B4 for the three-row calc method, the entry is invalid.<br><br>If you highlighted the full range of the calc method in the sheet before selecting to edit the calc method, then you can click **Show location in the current spreadsheet selection**, and the cursor will move to the location where the value would be placed. (If you are defining a variable as part of creating a new calc method, then you always have the calc method selected.) |
| Required | Specifies whether the user is required to enter or select a value for this variable.<br>• If this check box is selected, then the user must specify a value for this variable in order to insert the calc method, or use it to overwrite another.<br>• If this check box is not selected, then the user can leave the variable blank. In this case, the corresponding cell will be blank when the calc method is inserted or used to overwrite.<br><br>**NOTE:** This option does not apply to Related Column Value or GUID variables. |
| Insert Only | Specifies when the variable should be displayed:<br>• If this check box is selected, then the variable is only displayed when a user inserts the calc method. It is excluded from the form during all other operations, including change (overwrite) and double-click. If all variables for the calc method are set to Insert Only, then the form only displays when inserting.<br>• If this check box is not selected, then the variable is displayed in all cases.<br><br>**NOTE:** This option does not apply to Related Column Value or GUID variables. |

# Change calc method rules

End users can replace an existing calc method in the sheet by applying a new calc method, thereby changing the calculations used on the row (or rows).

The following replacement rules apply when using the **Change Calc Method** feature. When creating calc methods, keep these rules in mind so that change operations will work as desired.

The change operation is evaluated on a cell by cell basis. Actions depend on what the new calc method (the calc method replacing the existing calc method) contains in each cell.

| If the new calc method contains: | The following action occurs: |
|---|---|
| A value or a string | The value or string in the new calc method is ignored, and the existing cell value is retained as follows:<br>• If the existing calc method contains a value, a string, or is blank, then the existing cell contents are retained as is.<br>• If the existing calc method contains a formula, the value resulting from the formula is retained by using Paste Special – Values. The formula is removed. |
| Formula | The formula in the new calc method overrides the existing cell contents. |
| Empty cell | The empty cell in the new calc method overrides the existing cell contents. |

**NOTE:** The normal change rules do not apply to calc method variables. If the new calc method has a variable defined, the variable value specified by the user always overrides the existing cell contents. However, if the user does not specify a variable value (if the variable is not required and the user leaves the entry blank, or if the variable is excluded from the change operation because it is set to **Insert Only**), then the normal change rules apply to that cell.

7

# Drivers

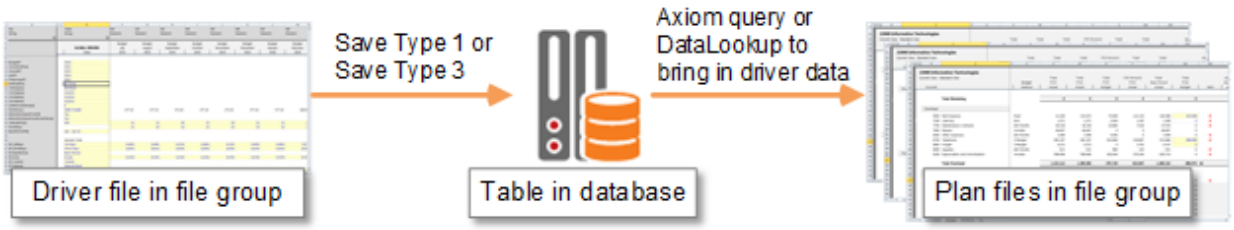
Driver files contain rates, statistics, and other drivers to be referenced in your plan files to help calculate planning data. Driver files can also contain global values such as calendar information. Driver files are defined on a per file group basis.

## About driver files

The data in driver files is ultimately saved to the database into one or more driver tables. These driver tables can then be referenced by templates/plan files to calculate planning data or determine other global settings for the file group. Each file group can have as many driver files and associated tables as needed.

### Driver tables

Although drivers are managed in spreadsheets, the driver data is ultimately saved to the database. When you use the GetData function to reference a driver value, Axiom is querying the associated table in the database, not the spreadsheet file.




Driver files can save data to the database using either of the following approaches:

- **Save Type 3**: Most driver files use Save Type 3 to create *document reference tables* in the database. When using this approach, the driver file is the ultimate source of the data. When the driver file is saved, the data is saved off to a document reference table. The document reference table cannot be edited separately; it can only be modified by editing and saving the source driver file.

- **Save Type 1**: Some driver files use Save Type 1 to save data to reference tables or data tables. When using this approach, the table is the ultimate source of the data. The data is queried into the driver file from the table, edited within the file, and then saved back to the table. The table can be edited separately from the driver file. Data should not be "stored" in the driver file; the queries in the driver file must be configured as rebuildable so that the driver file always has access to the latest data in the table.

When you create a new driver file, or add a sheet to an existing driver file, you must enable Save Type 3 or Save Type 1 for each sheet that contains data to be saved to the database.

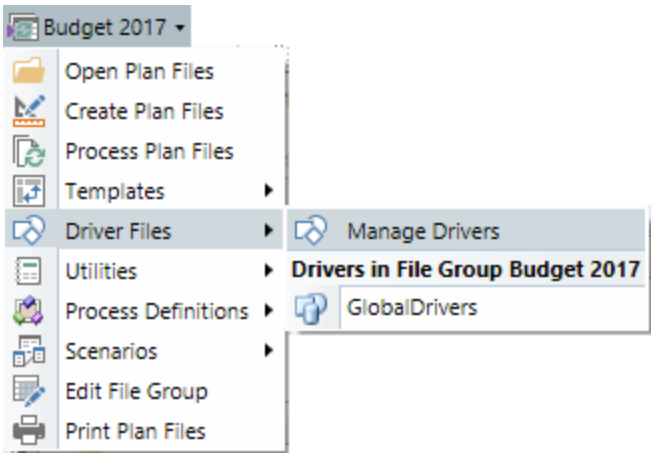
### Using multiple sheets or multiple driver files

Each file group can have multiple driver files, and each driver file can have multiple sheets. When deciding whether to create a new driver file or create a new sheet in an existing driver file, consider how the driver information will be used and which users need access to it. For example, if different business units in your organization have different sets of drivers, and each driver set is managed by a different user, you may want to create separate driver files instead of using multiple sheets in the same file.

Each driver sheet that you save to the database using Save Type 3 must use a unique table name. The table name does not have to incorporate the sheet name, but it is a good idea to do so, so that the relationship between the sheet and the table is clear.

# Managing driver files

You can manage the driver files for a file group by using the file group buttons on the **Axiom** tab, or by using the Explorer task pane. Clicking the arrow to the right of the file group name brings up the file group management menu.




By default, only administrators can create, edit, and delete driver files. Non-admin users can be granted permission to access driver files by using the **Files** tab in Security.

The **Manage Drivers** dialog offers a view of all your driver files and any associated document reference tables. This dialog has two viewing options:

- **Driver Documents**: View all of the driver documents for the file group.
- **Driver Tables**: View all of the tables associated with the driver documents. For each table, you can see the driver file that saves to the table, the sheet it is defined on, and the associated table variable (if applicable).

## Creating driver files

By default, all new driver files use the driver template defined for the system. In Axiom Explorer, the driver template file is located in `\Axiom\Axiom System\Document Templates\Drivers`. This file can be customized for the system if desired.

It is possible to define multiple driver templates for your system. If your system has multiple driver templates, then you will be prompted to choose a template when you create a new driver file. Only administrators can create new driver templates. The process of creating new driver templates is the same as for report templates, except that the driver templates must be saved in the Drivers template folder noted previously.

**To create a new driver file:**

1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Driver Files > Manage Drivers**.

   The **Manage Drivers** dialog opens, listing the driver files for the file group.

   > **TIP:** You can also access this dialog from the Explorer task pane by right-clicking the **Drivers** node and selecting **Manage Drivers**.

2. At the top right of the dialog, click **New**.
3. In the **Create File Group Driver** dialog, type a name for the new driver file into the **Name** box, then click **OK**.

   The name of the driver file can be anything you like. If the file group has multiple driver files, the name should be descriptive of the file contents, so that you can easily distinguish the driver files for editing purposes.

   For example, if you have one driver file for North American operations, and one driver file for European operations, you might name one driver file "Drivers_NorthAmerica" and one driver file "Drivers_Europe."

   The name of the driver file is not used when referencing driver data in templates / plan files. Driver data is ultimately saved to tables in the database, and these tables are used to reference the driver data.

4. If your system has multiple driver templates, select the template to use for the new driver file, and then click **OK**. If your system only has one template, then it will be used automatically.

A new driver file opens. You can now configure the driver file and add additional sheets as desired. See Setting up driver files.

New driver files can also be created in the following ways:

- You can create a new driver file by copying an existing driver file. If you are an administrator, you can use the Explorer task pane or Axiom Explorer to copy or import files into the Drivers folder of the file group (`\Axiom\File Groups\<FileGroupName>\Drivers`). You can also open the driver file that you want to copy, and then use **Save As (Repository)** to save it to a driver folder where you have read/write permissions.
- You can copy or import a report file into a Drivers folder. The file type is converted to driver as part of the copy or import.

## Editing driver files

You can edit a driver file at any time. If you edit the file, you must perform a full save in order to update the associated driver tables for the changes.

**To open a driver file for editing:**

- On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu. Click **Driver Files**, then click the name of the specific driver file. (Alternatively, click **All Drivers** to select a driver from the **Driver Files** dialog.)

  **TIP:** You can also open a driver file from the Explorer task pane or Axiom Explorer.

The driver file opens for editing.

## Deleting driver files

Deleting a driver file automatically deletes any document reference tables created from the file by use of Save Type 3. However, if the driver file uses Save Type 1 to save to a target reference or data table, the target table is not deleted.

**To delete a driver file:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The Axiom Explorer dialog opens, with the focus on the **File Groups** section of the treeview.

   **TIP:** You can also use the Explorer task pane to delete a driver file.

2. Expand the desired file group, and then select or expand the **Drivers** node.
3. Right-click the driver file and then select **Delete**.

# Setting up driver files

Driver files are used to gather user inputs and perform calculations, to generate the driver data to be used by the plan files in the file group. The driver data is saved to the database from the driver file, and then referenced in templates / plan files.

Driver files can use any Axiom file feature to query data into the file, present that data to users, and then save data to the database. You can design the file as needed to facilitate the necessary user inputs and calculations. Driver files can be designed for use as spreadsheet files or as web-enabled form files.

## Adding new sheets to driver files

You can add new sheets to driver files at any time using spreadsheet functionality. Most sheets in driver files are intended to be used for data entry, so that the data in the sheets can be saved back to the database using either Save Type 3 or Save Type 1.

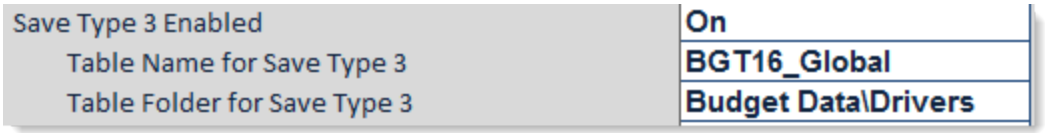
You can also add "supporting" sheets to the driver file, to be used for notes or to query data to be referenced by other sheets.

## Setting up a driver sheet to save to the database

In order to save data to the database, driver sheets must be set up for either Save Type 3 or Save Type 1.

Save Type 3 creates a document reference table, where the table is managed by the source spreadsheet. Each sheet to be saved must meet the following requirements:

- Save Type 3 must be enabled on the Control Sheet for that sheet.




- The sheet must be structured according to the rules of Save Type 3.
- Column titles must follow database column naming requirements.

Alternatively, driver files can use Save Type 1 to save data to an existing reference table or data table. In this case, Save Type 1 is set up as normal. When using Save Type 1 in a driver file, the driver file must use rebuildable data queries so that the primary source of data is the target table, not the driver file. Data should be queried from the target table to the driver file so that the user can make edits to the data, then the data should be saved back to the target table. The next time a user opens the driver file, the data query should be rebuilt with the latest data from the table, and so on.

For more information on setting up Save Type 1 or Save Type 3, see the *Axiom File Setup Guide*.

In both cases, the driver table name should be sourced from the file group variables instead of "hard-coded" within the driver file. For more information, see Driver table names and file group variables.

### File group variables

You can use the GetFileGroupID / GetFileGroupProperty / GetFileGroupVariable functions within the driver file to bring in the current file group ID, file group variables, and other file group properties. For example, you can use the GetFileGroupVariable function to bring in the appropriate tables to query data from and to save data to. This is very useful when cloning the file group for a rollover—the drivers can be updated along with the other files in the file group, keeping all related files in sync.

### Using calc method libraries in driver sheets

You can use calc method libraries within driver sheets. You might use this feature so that users with edit rights to the drivers can automatically insert predefined rows or sets of rows that define specific types of driver data. All of the calc method features that are available in templates/plan files are also available in driver files.

**NOTE:** When defining calc methods for a driver file, be very careful not to duplicate any calc method library names used in templates/plan files (or in utility files). Remember, calc method libraries are defined by sheet name and are used by all sheets in the file group with the same name. Your driver files and plan files likely do not share the same column structure and therefore should not share the same calc method libraries.

## Driver table names and file group variables

When configuring Save Type 3 or Save Type 1 for a driver file, the driver table name should meet the following requirements:

- Each save process should use a unique table name. This is an absolute requirement for Save Type 3, and a recommendation for Save Type 1.
- The table names should include some indicator of the file group that it belongs to.
- The table names should be defined as file group variables instead of "hard-coded" within the driver file. The GetFileGroupVariable function can be used to return the appropriate table name for each save process.

### Driver table names

When using Save Type 3, each driver sheet that is set up to save to the database must have a unique table name for the Save Type 3 configuration. The name must be unique not only within the file (meaning each sheet must save to a unique table name), but it must be unique across the Axiom system. If two sheets in the same file save to the same table name, or if two different files in the system save to the same table name, then the last sheet or file that is processed will overwrite the previous instance of the table.

For example, imagine that you have two driver files, one for your North American business units and one for your European business units. Both files have a Statistics sheet that you want to save to the database. Each of these sheets must be assigned a unique table name in the Save Type 3 settings, such as StatisticsNA and StatisticsEU.

When using Save Type 1, it is recommended to use a unique table name for each save process in the drivers, although it is not a requirement in this case. Using unique table names for each distinct sheet of data in the drivers makes it easier to set up the save processes and keep track of which data is being saved where. However, if it is appropriate for your system design, you can have multiple Save Type 1 processes in the same driver file or in different driver files that save to the same target driver table. To continue the previous example, you could have one Statistics table where you save the North American statistics from one driver file and the European statistics from another driver file. The data would have to share the same structure and include a grouping column where each row of data was identified as belonging to either North America or Europe.

Additionally, because file groups and their driver files are often cloned to create new file groups, it is recommended to dynamically construct the driver table names using some indicator of the file group that it belongs to. That way when the file group is cloned, new driver tables will be created for the new file group and the old tables will be left as is for use in the old file group. For example, the full table names from the previous example could be something like BGT21_StatisticsNA and BGT21_StatisticsEU, where the drivers belong to the Budget 2021 file group. The best way to enforce this dynamic naming convention is to use file group variables to define the driver table names.

## Using file group variables

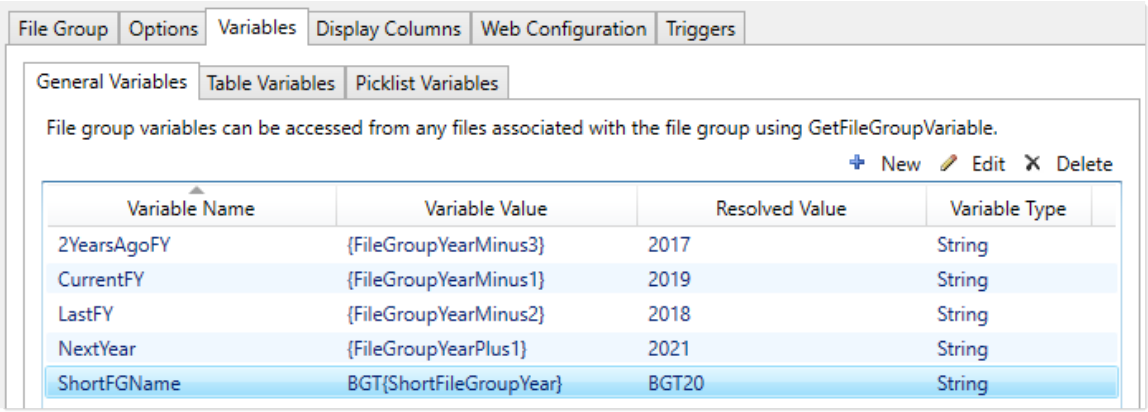
Ideally, your driver table names should be defined using table variables for the file group. You can then reference the table variables in the save-to-database settings using the GetFileGroupVariable function. Use of table variables allows you to manage these tables within the file group properties and also enables rapid development of planning scenarios using the file group scenarios feature. For more information, see the following topics:

- File Group Variables
- File Group Scenarios

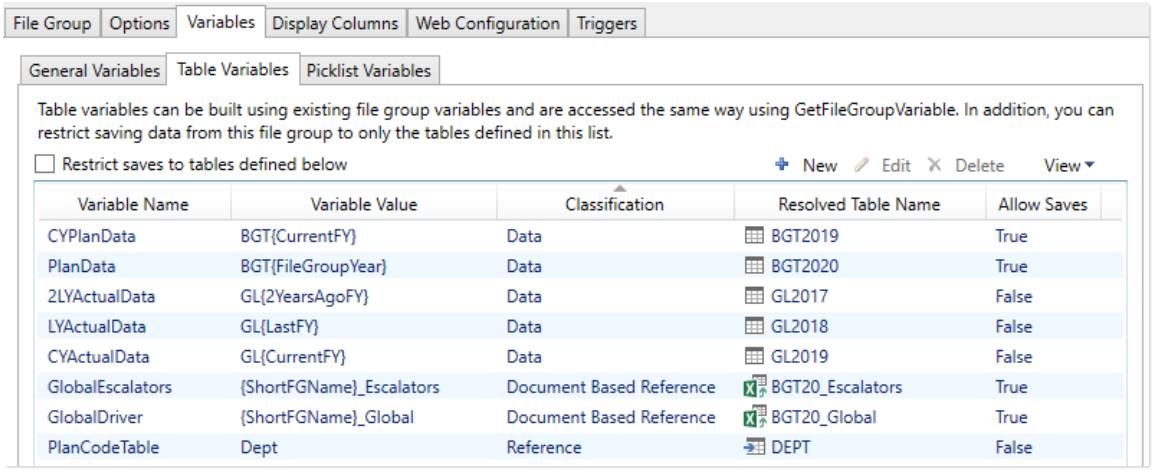
**NOTE:** If you have enabled the file group option to restrict saves to the target tables of table variables, then the designated table name in the save-to-database settings must match a table listed in the **Table Variables** tab for the file group. If it does not, the save-to-database process will fail.

The following is an example of how you can use table variables to create dynamic driver table names, and then reference those variables in the Save Type 3 settings:

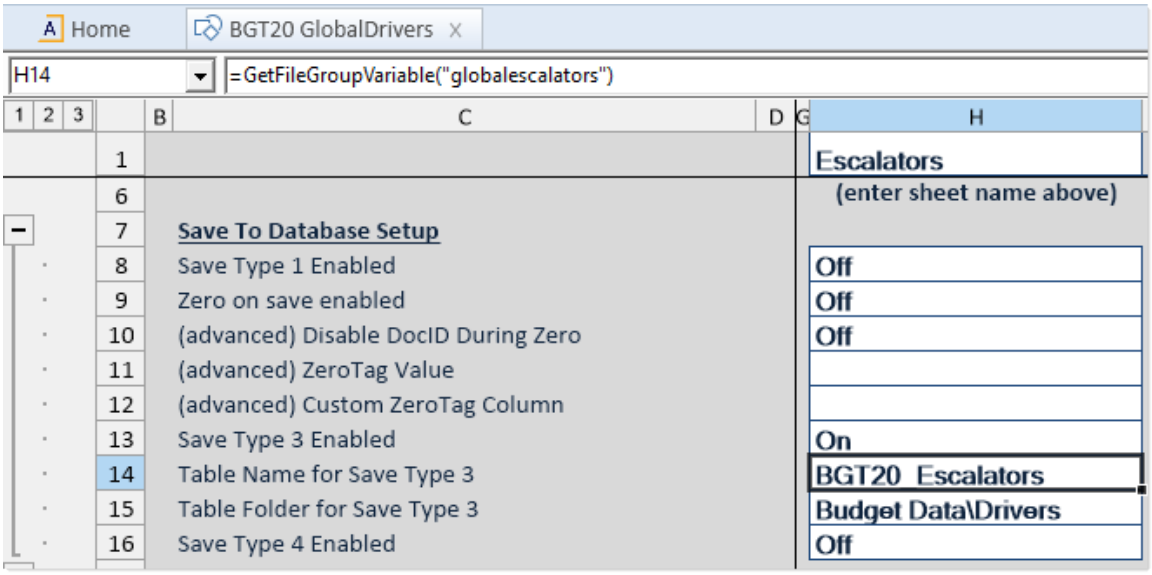
- In the file group settings, you can create a file group variable such as `{ShortFGName}`. This variable could reference the short version of the file group year and resolve to something like BGT20 (for a file group named Budget 2020). This variable could be used in several places, such as in the file group tab prefix, as well as in table variables for your drivers.




- When defining the table variables for the driver tables, use the ShortFGName variable as part of the table variable. For example, a driver table that holds escalator values might be a table variable named GlobalEscalators. This variable is defined as `{ShortFGName}_Escalators`, which resolves to BGT20_Escalators.



- In the Save Type 3 settings for the driver file, to define the table name, use the function GetFileGroupVariable to return the value for the GlobalEscalators variable.




Now the table name is dynamic, and it will change as the file group's variables change. If this file group is cloned and the year is changed to 2021, then the table name will resolve to BGT21_Escalators to keep the table name unique.

**NOTE:** The same approach applies to Save Type 1 driver files. You can use the GetFileGroupVariable function to return the expected name of the table, and then reference that name in the Save2DB tag to determine the destination table for the save-to-database.

# Processing driver files

Your driver files may be set up using Axiom queries and GetData functions that need to be updated regularly to provide the most current driver data to plan files. Rather than opening, refreshing, and saving each driver file, you can use Scheduler to process a set of driver files.

In the Scheduler **Process Document List** task, you define a set of files to process. You can opt to process Axiom queries and to perform a save-to-database as part of the task. You can then schedule the job for periodic processing, or you can run the job on demand.

For more information on Scheduler, see the *Scheduler Guide*.

# Referencing driver values in Axiom files

You can reference driver values in other Axiom files using any data query method: Axiom queries, data lookups, or GetData functions. For example, templates / plan files typically contain a Variables sheet that is used to query in driver values to be referenced throughout the file.

Axiom queries or data lookups are often the best way to bring driver values into a plan file. The values can typically be queried once when the file is opened, and then it is not necessary to update the values again during the user session. Using GetData functions to return driver values should be avoided unless you require the value to be continually updated.

## Understanding the Save Type 3 structure for data queries

When using Save Type 3 to create driver files, it is not possible to open the actual document reference table to see the table structure. You must understand how the Save Type 3 sheet structure translates to the eventual table structure, so that you can reference the table correctly in data queries.

For example, imagine that you have a driver sheet named PRDrivers that saves data to the BGT17_ PRDrivers table. Column A of the sheet defines the key column of the table, and contains the key values. Row 1 of the sheet contains the column names.

| | A | B | C |
|---|---|---|---|
| 1 | LookupKey | CYDriver | NYDriver |
| 2 | String | String | String |
| 3 | 50 | 30 | 30 |
| 6 | FICAMax | 106,800 | 106,800 |
| 7 | FICARate | 7.65% | 5.65% |
| 8 | FUTAMax | 7,000 | 7,000 |

To return the value for the FICA rate, a GetData data lookup would be set up as follows:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 6 | | | | | | | | |
| 7 | | | [DataLookup] | [result] | [iserror] | [columnname] | [tablename] | [filtercriteria] |
| 8 | | | [GetData] | 7.65% | | CYDriver | BGT17_PRDrivers | LookupKey='FICARate' |

Where "CYDriver" is the column that contains the value to retrieve, and "FICARate" is the key value in Column A of the driver sheet (`LookupKey='FICARate'`).

Remember, the data lookup is querying the table, not the spreadsheet, but when using Save Type 3 the spreadsheet is the only way to view the eventual table structure.

# Editing and saving driver values

You can edit the values in a driver file and save those values to the database, so that the values are available to plan files in the file group. When you save a driver file, the following occurs:

- The driver file is updated in the Axiom file system.
- Data from the file is saved to the Axiom database.

**To save a driver file:**

- On the **Axiom** tab, in the **File Options** group, click **Save**.

Before saving data to the database, Axiom performs an Excel calculation to update all formulas in the sheet (including any Axiom functions), and validates the file. If errors are found, the file is still saved but the data save is stopped and errors are displayed in the **Save Errors** pane. These errors must be corrected before data can be saved to the database. If no errors are found, a confirmation message displays information about the number of records saved.

The save-to-database process only occurs if the file is configured to save data to the database. However, most driver files are configured to save to the database.

## Additional save options

When you click **Save**, Axiom automatically performs all save actions for the file. This is the default save behavior that should be used in the majority of cases. If necessary, you can use additional save options to only save the driver file, or to only save the data. For example, you may be in the process of changing the structure of the driver file, and you don't want to save the data to the database until you have finished making changes. You can save just the file while you are working, and then when you are finished, perform a normal save.

Keep in mind that GetData functions reference tables in the database, not the driver files themselves. If you change the data in a driver file but do not save the data to the database, then plan files will not have access to the latest data.

**IMPORTANT:** If you save a driver file without performing a save-to-database (or vice versa), this may cause a mismatch between the data in the file and the data in the database. Ideally, the driver file save and the save-to-database should occur together, so that the database and driver files remain in sync. This primarily applies to driver files that use Save Type 3 to create document reference tables. When using Save Type 1, the driver files should be designed as rebuildable and therefore it is only necessary to save the data.

**To save only the file:**

- In the **Axiom** tab, in the **File Options** group, click the down arrow to the right of the **Save** button, and then click **Save File Only**.

The driver file is saved. All save-to-database processes are ignored.

**To save only the data:**

- In the **Axiom** tab, in the **File Options** group, click the down arrow to the right of the **Save** button, and then click **Save Data Only**.

Data from the file is saved to the database. The file itself is not saved.

8

# File Group Utilities

Each file group can have one or more utilities to perform associated functions for the file group. Utilities can be used for various purposes, such as:

- To calculate and save allocations for the file group
- To serve as an "add file form" dialog, to create new plan files for an on-demand file group
- To provide content for plan files—either as spreadsheet composite plan files, or as form-enabled plan files that use embedded forms
- To define step ownership assignments for a plan file process
- To define refresh variables for file group web pages, such as the Time in Step page and the Plan File Directory page

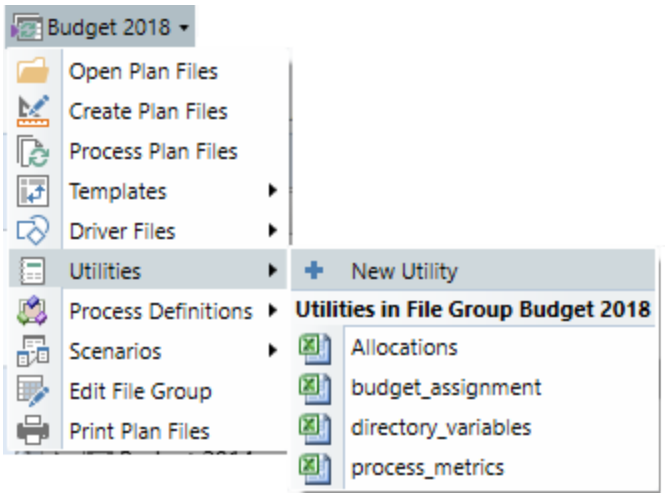
Utilities are Axiom report files that belong to a file group. These utilities can be accessed from the file group menu, can use file group features such as file group variables and calc method libraries, and can be included as part of file group cloning. And because utilities are report files, all the features available for reports are also available in file group utilities.

Utilities are entirely free-format. Typically your Axiom consultant will create these utilities as needed as part of the initial system development or other project work. You can create your own utilities at any time as well.

**NOTE:** It is possible to store other types of files in the Utilities folder of a file group, so that these files belong to the file group and can leverage that association. For more information, see Using other file types as file group utilities.

## Managing utilities

You can manage the utility files for a file group by using the file group buttons on the **Axiom** tab, or by using the Explorer task pane. Clicking the arrow to the right of the file group name brings up the file group management menu.




By default, only administrators can create, edit, and delete utility files. Non-admin users can be granted permission to access driver files by using the **Files** tab in Security.

Within the Axiom file system, file group utilities are stored in the `\Utilities` sub-folder within the file group folder.

## Creating utility files

By default, all new utility files use the utility template defined for the system. In Axiom Explorer, the utility template file is located in `\Axiom\Axiom System\Document Templates\Utilities`. This file can be customized for the system if desired.

**To create a new utility file:**

- On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Utilities > New Utility**.

  **TIP:** You can also create new utilities from Axiom Explorer or the Explorer task pane, by right-clicking the Utilities node and selecting **New > Utility**.

A new utility file opens. You should save this file before starting to configure it.

New utility files can also be created in the following ways:

- You can create a new utility file by copying an existing utility file. If you are an administrator, you can use the Explorer task pane or Axiom Explorer to copy or import files into the Utilities folder of the file group (`\Axiom\File Groups\<FileGroupName>\Utilities`). You can also open the utility file that you want to copy, and then use **Save As (Repository)** to save it to a utility folder where you have read/write permissions.
- You can copy or import an existing report file into a Utilities folder, or save a copy of an existing report to a Utilities folder. The **Save As** dialog for reports allows saving report files to any utilities folder that you have access to, as well as the normal Reports Library folders.

## Editing utility files

You can edit a utility file at any time. Keep in mind that utility files may be actively used by various processes. For example, a utility file that serves as an assignment workbook for an active plan file process will be continuously accessed as needed to look up ownership assignments. If you are editing such a file, you should take care not to save the file with "in progress" changes that have not yet been completed. If necessary, you may want to create a copy of the file to develop your changes, and then overwrite the existing file with the new file when changes are complete.

**To open a utility file for editing:**

- On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu. Click **Utilities**, then click the name of the specific utility file.

  **TIP:** You can also open a driver file from the Explorer task pane or Axiom Explorer.

The utility file opens for editing.

## Deleting utility files

Before deleting a utility file, you should be sure that the file is not being used by any processes or configuration settings.

**To delete a utility file:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The Axiom Explorer dialog opens, with the focus on the **File Groups** section of the treeview.

   **TIP:** You can also use the Explorer task pane to delete a utility file.

2. Expand the desired file group, and then select or expand the **Utilities** node.
3. Right-click the utility file and then select **Delete**.

## Automated processing of utilities

If you need to schedule processing of a file group utility, you can use either of the following Scheduler tasks (depending on whether the file is set up for File Processing or not):

- Process Document List (to refresh queries and save-to-database without using File Processing features)
- File Processing (to perform File Processing actions)

It is also possible to iteratively process utilities per plan file, where each utility is processed using data unique to the current plan file. This is typically used in conjunction with form-enabled plan files that use embedded forms, but it can also be used with regular plan files as needed. For more information, see Using utility processing for plan files.

# Setting up utility files

Utility files can be used for just about any purpose. Utility files are the same as report files, except that utility files are associated with a file group.

Utility files can use any Axiom file feature that reports can, as well as additional features that can only be used within file groups, such as file group variables and calc method libraries. Utility files can be designed for use as spreadsheet files or as web-enabled form files.

### File group variables

You can use the GetFileGroupID / GetFileGroupProperty / GetFileGroupVariable functions within the utility to bring in the current file group ID, file group variables, and other file group properties. For example, you can use the GetFileGroupVariable function to bring in the appropriate tables to query data from and to save data to. This is very useful when cloning the file group for a rollover—the utilities can now be updated along with the other files in the file group, keeping all related files in sync.

### Calc method libraries

You can use calc method libraries within the utility. For example, in an allocations utility, you might want to use different calc methods for different departments or accounts. This is a simple matter when using a calc method library.

All of the calc method features that are available in plan files and driver files are also available in utilities.

**NOTE:** When defining calc methods for a utility, be very careful not to duplicate any library names used in plan files. Remember, calc method libraries are defined by sheet name and are used by all sheets in the file group with the same name. Your utility files and plan files likely do not share the same column structure and therefore should not share the same calc method libraries.

# Using utility processing for plan files

You can configure plan file templates to enable "plan file processing with utilities" instead of traditional plan file processing. This means that when plan files are processed, a list of utilities is iteratively processed per plan file, using data for that plan code. This feature is primarily intended to be used with form-enabled plan files that use embedded forms, where the content of the form plan files is comprised of multiple embedded utility files instead of within the plan file itself.

For example, imagine that you have a plan file that uses Utility1, Utility2, and Utility3 as embedded forms for content. When users are working in the form plan file, they can switch between the embedded utilities by using a Menu component. They can input values in each utility and save data to the database. The embedded utilities are filtered to only show and save data for the current plan code, by use of shared variables.

Now imagine that a driver value in the file group has changed, so you want to process plan files to save updated data to the database. You cannot use the regular Process Plan Files feature for this purpose, because it only processes and saves the plan file. In this example, the plan file itself is only a "shell" that provides context for the embedded utility files. The utility files are where all of the data processing occurs for the plan code, so it is the utility files that need to be processed and saved. Therefore in order to process these plan files, you must perform iterative processing of utilities per plan code. In this example, this means that Utility1, Utility2, and Utility3 would be processed repeatedly—once for each plan code—in order to save the updated data for each plan code to the database.

Although plan files with embedded forms are the intended use case for this feature, "processing with utilities" can be used with any kind of plan file where regular utility processing per plan code is required. However in most other cases, using regular multipass file processing with the utility is usually sufficient, and easier to set up.

## Setting up plan file utility processing

In order to use plan file processing with utilities, the plan file templates and the utility files must be set up as follows:

- **Define the data source for utility processing.** The plan file template (and therefore the resulting plan files) must contain a ProcessPlanFileUtilities data source. This data source lists the utility files to be processed, and the processing order. If the list of utilities varies per plan file, formulas can be used to dynamically enable and disable utilities as needed. Document variables can also optionally be passed to each utility for processing. For more information, see Creating ProcessPlanFileUtilities data sources.
- **Use variables to pass the current plan code.** Either *shared variables* or *document variables* must be used to pass the current plan code (and any other necessary values) from the plan file to the utility files. Shared variables can also be used to pass information from one utility to another utility that is later in the processing order.

  For plan files with embedded forms, the template and the utility files are already set up to use shared variables, because this is how the files share values in the forms environment. If you use utility processing with other types of plan files, then you can either choose to set up shared variables in the files just for utility processing, or you can choose to pass the plan code to each utility using document variables.

- **Filter utilities by current plan code.** The utility files must be set up to filter data queries as needed for the current plan code being processed (and any other necessary values). Utility processing does *not* automatically filter the utility files by the current plan code.

  You must manually set up the filters as needed using either shared variables or document variables.

  - When using shared variables, at minimum the template must contain a SetSharedVariable function to set the plan code value, and the utilities must contain a GetSharedVariable function to return the plan code value.
  - When using document variables, at minimum the ProcessPlanFileUtilities data source in the template must be set up to pass the plan code value as a document variable, and the utilities must contain a GetDocumentInfo function to return the plan code value.

  Again, when using plan files with embedded forms, the utilities are already set up to be filtered by the current plan code using shared variables, as that is how the utilities show the appropriate values in the forms environment.

## Executing plan file utility processing

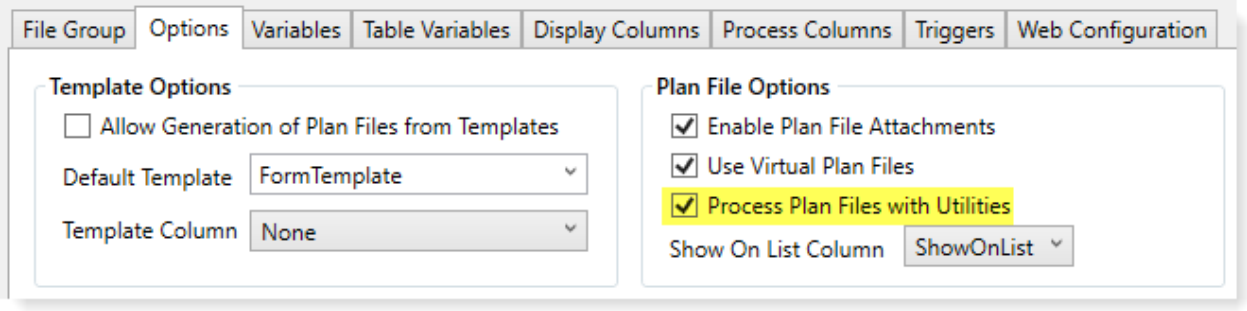
To execute plan file processing with utilities, use the **Process Plan Files** feature with the **Process with Utilities** option. You can select the plan files to process, and select which ProcessPlanFileUtilities data source to use (if there are multiple). For each plan file to be processed, the following occurs:

- The plan file is opened and refreshed, and shared variables are set.
- Each utility is opened and processed as follows:
  - Shared variables and document variables are passed to the utility.
  - Data queries are refreshed.
  - A save-to-database is performed.

For more information, see Processing plan files with utilities.

## Enabling plan file utility processing as the default processing mode

If you have a file group where utility processing is the primary means of plan file processing, you can configure the file group so that it is the default processing mode. In the file group properties, on the **Options** tab, enable **Process Plan Files with Utilities**. Typically this would only be done for file groups that use form-enabled plan files with embedded forms.




Enabling this setting has the following impacts:

- When using Process Plan Files with the file group, the **Process with Utilities** option is selected by default.
- For plan file process definitions, if a step is configured to **Save and validate plan file before advancing to next step**, then utility processing will be executed for the plan file instead of executing a save-to-database on the plan file itself. A ProcessPlanFileUtilities data source must be flagged as the ProcessValidation data source in the plan file to be used for this purpose (or else the default data source is used).

## Creating ProcessPlanFileUtilities data sources

If your plan files need to use plan file processing with utilities, the plan file template must contain a ProcessPlanFileUtilities data source. This data source specifies the utilities to be processed and the processing order. You can also optionally pass one or more document variables to the utilities to use during processing.

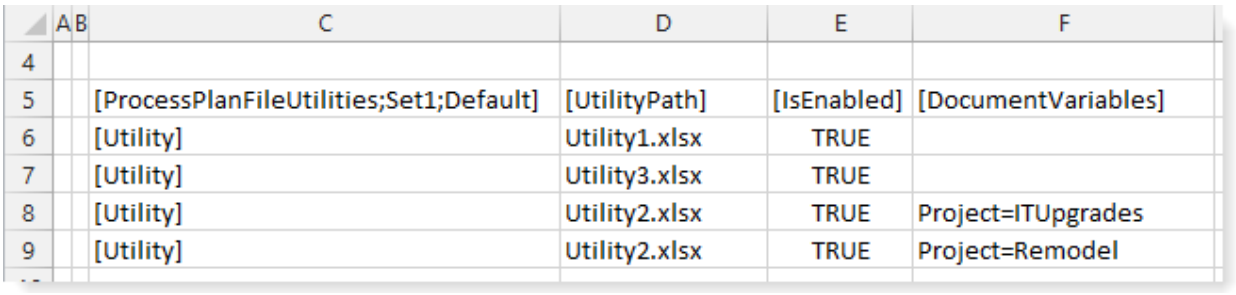
A plan file template can contain one or more ProcessPlanFileUtilities data sources as needed. When using Process Plan Files, you can select the data source to use for processing. When performing utility processing as part of plan file process step validation, one of the data sources must be marked as the data source to use for processing.

**To create a ProcessPlanFileUtilities data source in a template:**

- Right-click the cell in which you want to start the data source, then select **Axiom Wizards > Insert Process Plan File Utilities Data Source**. The current cell and any necessary cells below and to the right must be blank in order to insert the data source. This option is only available in template files.

The primary tag, column tags, and a row tag are placed in the sheet. From here, you can add more row tags and complete the column contents as needed.

The following screenshot shows an example ProcessPlanFilesUtilities data source. In this example Utility1 is processed first, followed by Utility3 and then Utility2. Utility2 is processed twice, using different values for the document variable Project.



| | C | D | E | F |
|---|---|---|---|---|
| 4 | | | | |
| 5 | [ProcessPlanFileUtilities;Set1;Default] | [UtilityPath] | [IsEnabled] | [DocumentVariables] |
| 6 | [Utility] | Utility1.xlsx | TRUE | |
| 7 | [Utility] | Utility3.xlsx | TRUE | |
| 8 | [Utility] | Utility2.xlsx | TRUE | Project=ITUpgrades |
| 9 | [Utility] | Utility2.xlsx | TRUE | Project=Remodel |

*Example ProcessPlanFileUtilities data source*

The ProcessPlanFileUtilities data source uses the following tags:

### Primary tag

**`[ProcessPlanFileUtilities;`*`DataSourceName`*`;`*`DataSourceType`*`]`**

The *DataSourceName* identifies this data source. When running Process Plan Files, you specify the data source to use by choosing the data source name. The name is required even if the template only has one data source.

The *DataSourceType* is optional, and can be used to flag a data source as one of the following types:

- `Default`: Flags the data source as the default data source for use with Process Plan Files. The default data source will be used for processing unless you select a different data source.
- `ProcessValidation`: Flags the data source to be used when validating the plan file as part of completing a plan file process step. This data source is used when **Process with Utilities** is enabled for the file group, and **Save and validate plan file before advancing to next step** is enabled for the step. If no data source is flagged as the ProcessValidation data source, then the default data source is used.

The placement of this primary tag defines the control column and the control row for the data source.

- All column tags must be placed in this row, to the right of the tag.
- All row tags must be placed in this column, below the tag.

### Row tags

**`[Utility]`**

Each row flagged with this tag specifies a utility file to include in the processing. Utilities are processed in the order listed in the data source, from top to bottom.

### Column tags

**[UtilityPath]**

The file path and name of the utility to be processed. If only a file name is listed with no path, the location is assumed as the `\Utilities` folder for the current file group.

For example, you can list a file as follows:

`\Axiom\File Groups\Capital\Utilities\Utility1.xlsx`

`Utility1.xlsx` (if the file is in the root of the Utilities folder)

`SubFolder\Utility1.xlsx` (if the file is in a subfolder of the Utilities folder)

If you choose to list the full path, it is recommended to use formulas to construct the path dynamically, so that the path will update if the file group is cloned and the utilities are copied to the new file group. For example, you can use the `GetFileGroupProperty ("UtilitiesPath")` function to return the path to the Utilities folder for the current file group.

**[IsEnabled]**

Specifies whether the row is enabled for processing (True/False). By default, blank is interpreted as False, so you must enter True if you want the utility to be processed. If a row is flagged as False, that row is skipped for processing.

This column can be used to enable dynamic processing of utilities, so that a utility can be processed or not based on a condition. The data source is evaluated separately for each plan file being processed, so a formula could be used that resolves to True for some plan files, and False for others.

**[DocumentVariables]**

Defines one or more document variable / value pairs to pass to the target utility. Variable / value pairs are specified as follows:

```
Variable1=Value;Variable2=Value
```

Separate multiple variable / value pairs using semicolons. If a value contains a semicolon, then it must be preceded by a backslash (\) so that Axiom does not treat the semicolon as a delimiter. Equals signs within a value (such as to pass a filter criteria statement as a value) do not need to be specially treated.

When the utility is opened for processing, the designated document variables are passed to the target utility and can be returned in that file using GetDocumentInfo functions. For example, if the `[DocumentVariables]` column contains Project=Remodel, that value can be returned within the utility by using the following function:

```
=GetDocumentInfo("Variable","Project")
```

You might use document variables if you want to process the same utility repeatedly for the same plan file, using different values each time the utility is processed. The previous example screenshot processes Utility2.xlsx twice, using different values for the Project variable.

You might also use document variables if you are using utility processing with regular spreadsheet plan files, and therefore the plan files and utilities are not already set up with shared variables. In this case, document variables can be used as an alternative to pass the current plan code (and any other relevant values) into each utility file.

**NOTES:**

- The primary tag must be placed in the first 500 rows of the sheet.
- Formulas can be used to create the tags, as long as the initial bracket and identifying keyword are whole within the formula.

# Using other file types as file group utilities

In some cases, non-report file types can be stored in a file group Utilities folder, so that these files are associated with the file group and can use certain features that leverage this association. The following additional file types can be stored in the Utilities folder of a file group:

- Task panes
- Scheduler jobs

When task panes and Scheduler jobs are stored in a file group Utilities folder, they are copied as part of any file group cloning activity that includes copying utilities. The task panes and Scheduler jobs should be set up so that they will work as expected in the new file group context.

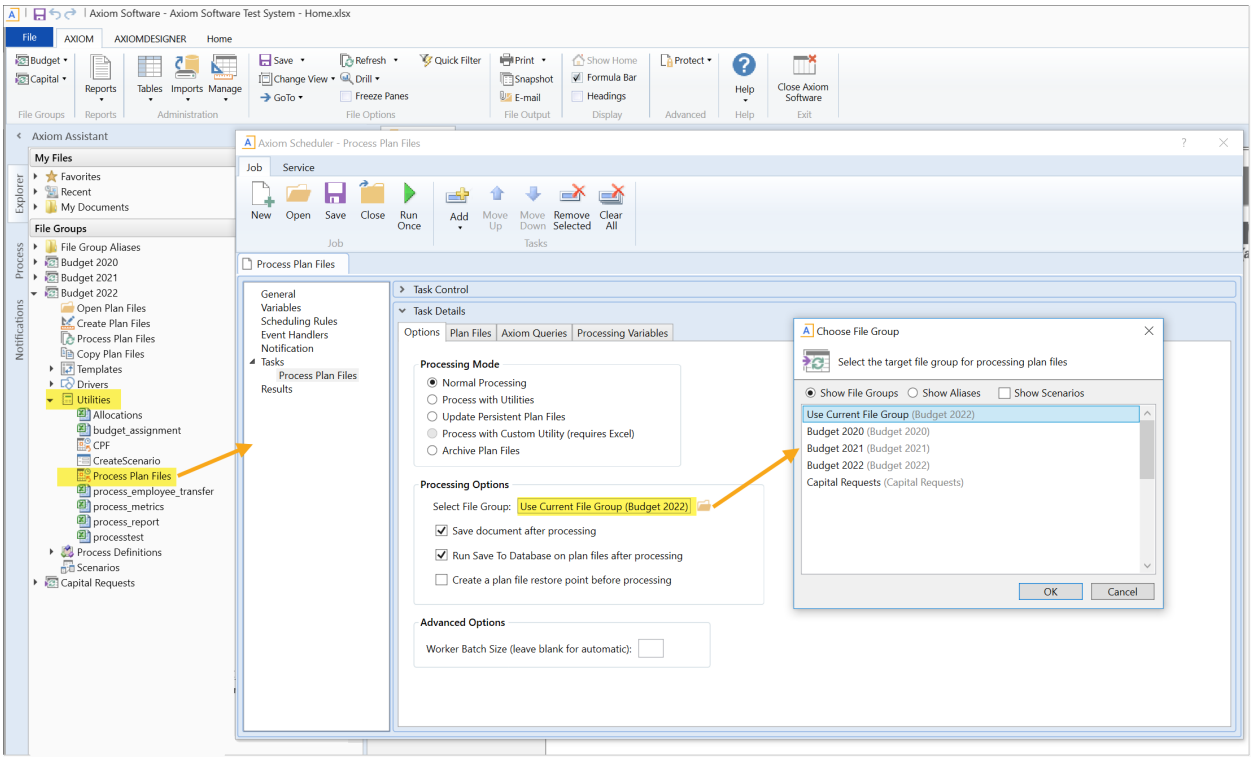
**NOTE:** Currently it is not possible to directly save a task pane or a Scheduler job to a file group Utilities folder when creating the file. You must first create the file and save it to the Task Panes Library or the Scheduler Jobs Library, then use Axiom Explorer to copy the file to the file group Utilities folder. You can also export the file and then import it into the Utilities folder.

### Scheduler jobs

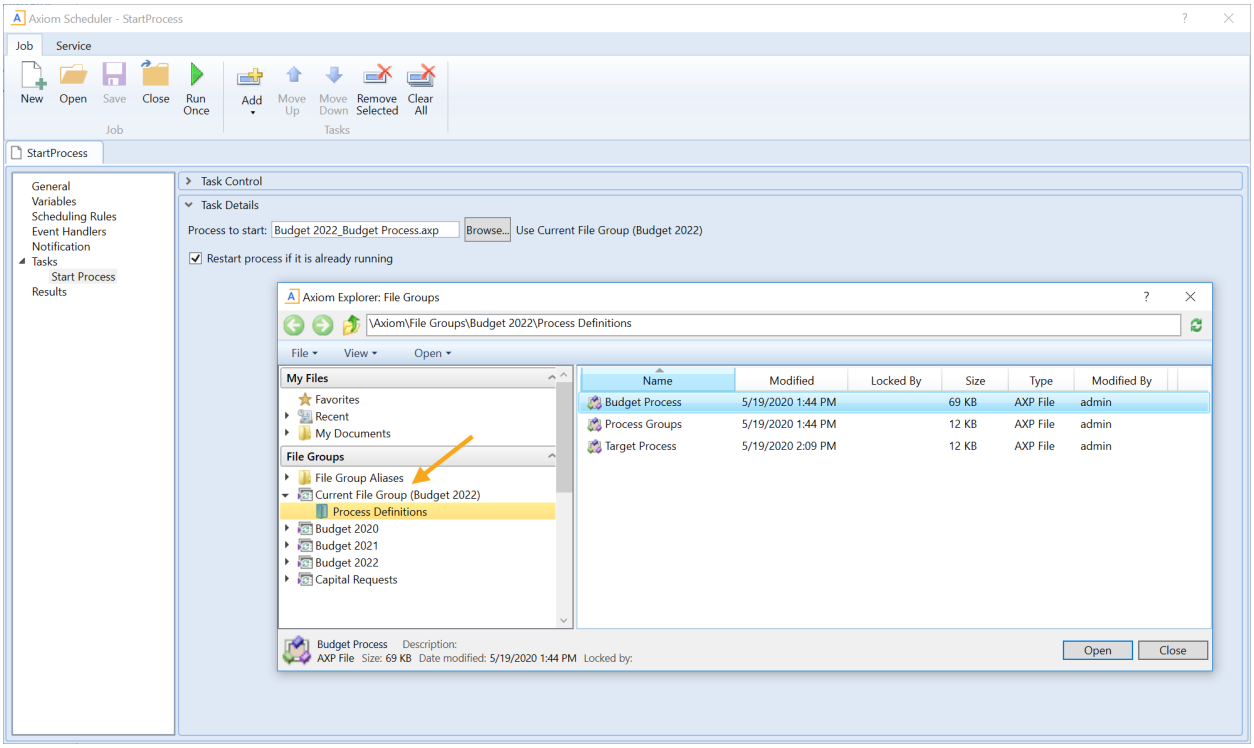
If a Scheduler job is stored in a file group Utilities folder, the following task types can be configured as relative to the current file group. This is the recommended way to configure the tasks, so that the tasks will automatically reference the current file group when the file group is cloned.

| Task | Notes |
|---|---|
| Create Plan Files | When selecting the file group for the task, you can optionally select **Use Current File Group** instead of selecting a specific file group or alias. |
| Execute Command Adapter (Create File Group Scenario command) | When selecting the file group for the command, you can optionally select **Use Current File Group** instead of selecting a specific file group or alias. |
| File Processing | When selecting the file to process for the task, you can optionally select a report utility file underneath the **Current File Group** node at the top of the file groups list. If you do this, the path to the file is stored relative to the current file group. |
| Process Plan Files | When selecting the file group for the task, you can optionally select **Use Current File Group** instead of selecting a specific file group or alias. |
| Start Process | When selecting the process definition file for the task, you can optionally select a file underneath the **Current File Group** node at the top of the file groups list. If you do this, the path to the file is stored relative to the current file group. |

In the following example, a Process Plan Files task is configured to **Use Current File Group**, which is currently Budget 2022. If this file group is cloned, or if a file group scenario is created for the file group, this Scheduler job will be copied with the other utilities and it will automatically point to the new current file group.

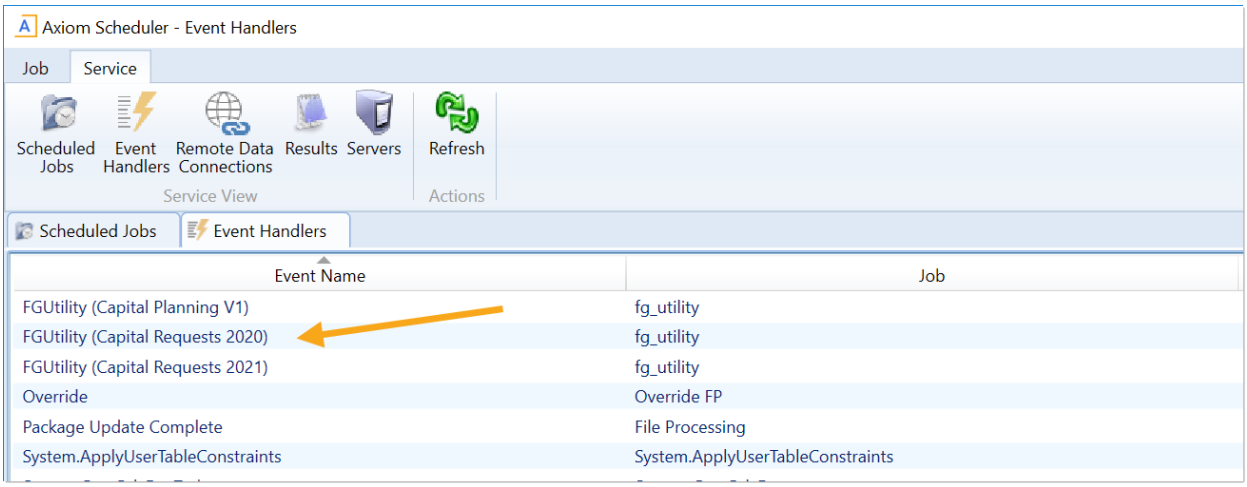



The next example shows how you can select a plan file process definition underneath the **Current File Group** node when configuring the **Start Process** task. If this file group is cloned, this Scheduler job will be copied with the other utilities and it will automatically point to the plan file process definition within the new current file group.



## File group-specific event handlers

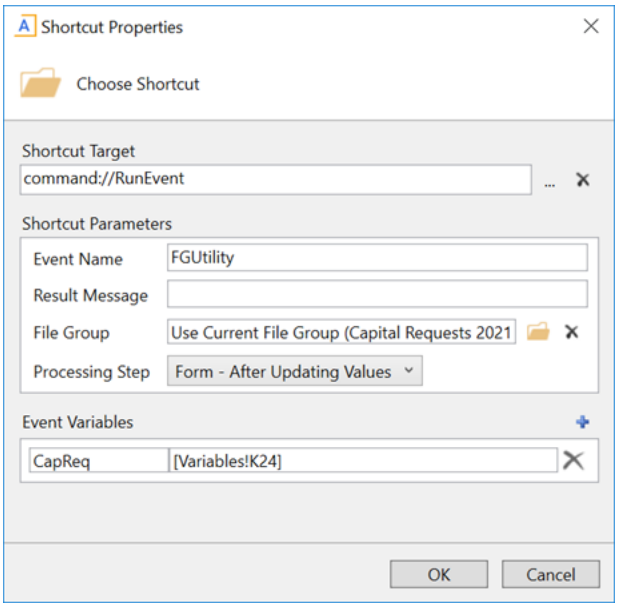
When a Scheduler job with an active event handler is stored in a file group Utilities folder, the event handler becomes specific to the file group. The **Event Handlers** tab in Scheduler displays the name of the associated file group next to the event handler name. If the file group and its utilities are cloned using any process—such as regular file group clone, create scenario, or file group rollover—then the event handler is copied for use in the new file group. If a file group and its utilities are deleted, the associated event handler will also be deleted.




*Example event handlers associated with a file group*

The RunEvent command can be configured to target event handlers stored in a specific file group. If a file group context is specified, then Axiom looks for the event handler name in that file group and in the general Scheduler Jobs Library, but not in any other file groups. If no file group context is specified, then Axiom looks for the event handler name in all locations. Note the following exceptions for related features:

- The RunEvent function does not have a parameter to specify a file group context. Instead, if the spreadsheet with the RunEvent function is stored in a file group, then that file group context is automatically applied. If the spreadsheet is not part of a file group, then no file group context applies.
- The Raise Event task in Scheduler does not support the ability to specify a file group context. It will always look for the event handler name in all locations.




*Example RunEvent command to trigger an event within a file group*

## Task panes

The primary use case for storing task panes in a file group Utilities folder is in support of composite plan files. Composite plan files are comprised of multiple parts, and an associated task pane can be used to allow users to open each of the associated parts. Currently, composite plan files can only be configured by product developers and delivered as part of a packaged product.

Task panes associated with a file group can also use the **Use Current File Group** option when configuring a RunEvent command to trigger a Scheduler job, or when configuring the Create File Group Scenario command to create a scenario.

9

# Plan File Management

This section discusses the setup and administration processes for plan files, including using the **Create Plan Files** utility to create plan files, refreshing plan files with data, and controlling user access to plan files.

For information on how end users work with built plan files to develop plans, see the *Desktop Client User Guide*.

## Assigning templates to plan codes

Before you can use the **Create Plan Files** utility to create plan files for a file group, you must first assign each plan code to a template. There are two file group features that you can use to assign templates to plan codes:

- **Template Column**: You can designate a column in the plan code table which holds the template assignments for each plan code. For example, the BudTemplate column could be designated to hold the template assignments for the budgeting process.
- **Default Template**: You can designate a template that will be used if no template column is specified, or if a template column is specified but a plan code does not have an entry in the template column.

Both of these settings are defined in the file group properties.

The template column is not automatically created in the plan code table. If you want to use a template column, it must be manually created. The template column can be named anything you like, but it must be a string column.

The easiest way to populate the template column with template names is to open the appropriate table using **Open Table in Spreadsheet**. When entering template names, note the following:

- Do not include the file extension on the template name. For example, enter just `Admin`, not `Admin.xlsm` or `Admin.xlsx`.
- You can optionally use file group variables in this column, enclosed in curly brackets. For example: `{AdminTemplate}`. When a template assignment needs to be determined, this variable will be resolved to the value defined in the file group settings.

A plan code table can be used by multiple file groups, and therefore can have multiple template columns. However, if the same template settings apply to multiple file groups, the same template column can be used.

For more information on creating columns in tables and using **Open Table in Spreadsheet**, see the *System Administration Guide*.

**NOTES:**

- On-demand file groups typically do not use the **Create Plan Files** utility. Instead, plan files are created by end users as needed. The Template Column and Default Template options still apply in this case and are used to determine the appropriate template for the new plan file. The exception is if an end user chooses to create a new plan file by cloning an existing plan file—in that case, the existing plan file is the "template." For more information, see Template options for on-demand file groups.
- When using virtual plan files, keep in mind that changing the template assignment for a plan file will not cause it to use that template the next time the plan file is accessed. Even though virtual plan files are rebuilt from template each time they are accessed, they are rebuilt using the original template. If you want the virtual plan file to use a different template, then you must use the Create Plan Files utility to "re-create" the placeholder document record for the virtual plan file, thereby associating it with the new template.

# Creating plan files

Using the **Create Plan Files** utility, you can create one or more plan files for a file group. Typically you would selectively create plan files during the setup and testing phase, and then you would create all plan files when you are ready to roll out the system to end users.

The Create Plan Files utility simply creates and saves the plan files—it does not refresh the files with data. Once you have created the plan files, run the **Process Plan Files** utility to populate Axiom queries with data.

In order to use the Create Plan Files utility, all plan codes must already be assigned to a template. For more information, see Assigning templates to plan codes.

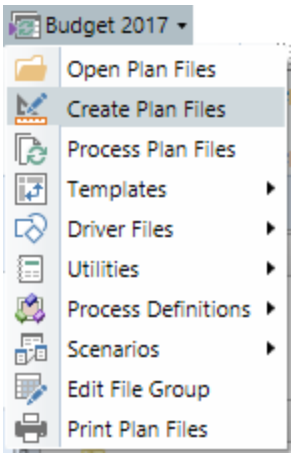
The Create Plan Files utility is only available to administrators and to users with the **Create Plan Files** security permission. Additionally, users must have read/write access to the plan files that they want to create.

**NOTES:**

- By default, all plan codes in the plan code table are available to the Create Plan Files utility. If some plan codes should be excluded from this utility, you can use a Show on List column. For more information, see Using a Show On List column.
- The Create Plan Files utility is only available for file groups with **Allow Generation of Plan Files from Templates** enabled in the file group properties. By default, this is enabled for standard file groups and disabled for on-demand file groups.
- If the file group uses virtual plan files, then the Create Plan Files utility creates the placeholder document record so that users can access the file. Physical plan files are not actually created by the utility in this case. For more information, see Using virtual plan files.
- You can also use Scheduler to create plan files, using the **Create Plan Files** task.

**To create plan files:**

1. On the **Axiom** tab, in the **File Group** group, click the arrow next to the file group name to bring up the file group menu, then click **Create Plan Files**.




**TIP:** You can also access this command in the Explorer task pane.

The **Create Plan Files** dialog opens, listing the plan codes for the file group. You can sort, filter, and group the list using standard Axiom grid features. This dialog always includes the following columns, which provide useful information for the plan file creation process:

- **File Exists**: Specifies whether a plan file already exists for this plan code (True/False). This can help you determine whether you will be overwriting an existing plan file.
- **Assigned Template**: Displays the name of the template that will be used to create the plan file. This assignment is taken from either the designated Template Column or the Default Template for the file group. If this entry is not as expected, then you should double-check the file group settings and the template column as appropriate.

2. Select the plan codes for which you want to create plan files. You can do this using one of the following methods:

   - **Create all plan files:** To create all plan files, select **All**.
   - **Create selected plan files:** To create certain plan files, select **Choose from list**, and then select the check boxes for those individual plan codes.

     To find the plan codes you are looking for, you can sort, filter, and group the list using standard Axiom grid features. You can show additional columns and hide columns by right-clicking in the column header. If you have filtered the list, you can select the check box in the header to select only the plan codes that currently display in the dialog.
   - **Create a subset of plan files using a filter:** To use a filter to create a subset of plan files, select **Use filter**, and then type a filter into the filter box. You can also use the Filter Wizard to build the filter. The filter must use the plan code table or a reference table that the plan code table links to. For example: `DEPT.Region='West'`.

     Once you have entered a filter, you can click **Refresh plan file list** to show only those plan codes that currently match the filter. The refresh feature is to help you determine whether you have defined the filter correctly.

   **IMPORTANT:** If you select a plan code that already has a plan file, the **Create Plan Files** utility will overwrite the existing file and create a new one.

3. Click **OK**.

   A message box displays the number of plan files you are about to create, and asks you to confirm that you want to continue. If you are overwriting any existing plan files, the message also informs you that a restore point will be created before the process begins, so that if necessary you can restore plan files that were overwritten (see Restoring plan files from restore points).

4. Click **Yes** to continue.

   If the number of plan files is not as expected, you can click **No** to cancel the process and return to the **Create Plan Files** dialog.

When the **Create Plan Files** process is complete, a dialog opens to display the results for each plan code included in the utility. The **Result** column displays success or failure, and the **Details** column provides additional information.

**NOTE:** If you are using Create Plan Files to create new on-demand plan files, those plan files will be automatically started in the designated **Plan File Process** for the file group. This only applies when creating a brand new plan file. If an existing plan file is overwritten, its process status will be left as is.

### After creating plan files from templates

Once plan files have been created from templates, further changes to a template do not impact plan files, unless the plan files are re-created. However, you should be careful not to make any significant changes to a template once plan files are created, because the template is referenced by Process Plan Files and Print Plan Files to obtain lists of Axiom queries and print views respectively. Therefore you should not rename, delete, or otherwise edit any Axiom queries or print views in a template unless you plan to re-create plan files, or unless the existing plan files are no longer active and will not be used by these utilities.

Generally speaking, the Control Sheets of individual plan files should not be edited. If a change needs to be made to the Control Sheet, the change should be made in the template and then plan files should be re-created. Most importantly, you should not edit the Axiom queries in an individual plan file, because this may cause unexpected results when updating the plan file using Process Plan Files.

**NOTE:** These considerations do not apply if the file group uses virtual plan files. Virtual plan files are automatically rebuilt from the template each time they are accessed, so the plan file always uses the latest version of the template.

### Using Create Plan Files with on-demand file groups

In most cases, the Create Plan Files utility is not used with on-demand file groups. By default, **Allow Generation of Plan Files from Templates** is disabled for on-demand file groups, so the utility is not available in the file group menu. Instead of creating plan files in bulk using the utility, users create plan files "on demand."

However, in rare cases, the utility can be used with on-demand file groups. For example, you might import records from another system and need to generate plan files for those records in Axiom. In that case, you can enable Allow Generation of Plan Files from Templates so that you can use the Create Plan Files utility to do so.

# Processing plan files

You can use the **Process Plan Files** feature to process plan files in batch. The following actions can be performed for each plan file:

- Refreshing data, including running selected Axiom queries
- Executing a save-to-database
- Saving the file

Process Plan Files is intended to be used by administrators and other power users to process multiple plan files at a time. This feature can be used throughout the planning cycle to update plan files and the corresponding data in the database. Typical uses of Process Plan Files include the following:

- After initially creating plan files, you can run Process Plan Files to build out the Axiom queries and save the initial planning data to the database.
- Plan files can reference information that is held in driver tables (and other reference tables). If you have updated these tables, the data in plan files may be out of sync with the planning data in the database. To synchronize the data, you can run Process Plan Files to refresh the data in plan files and save data to the database.

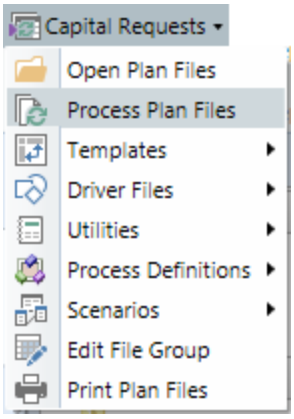
  For example, imagine that you change the value of an escalator from 2% to 2.5%. Calculations in plan files that reference this escalator value will now return a different result. In order to update the planning data in the database, you can process plan files to bring in the new escalator value, recalculate the data in the file, and then save the updated data to the database.
- If your plan files are designed to accommodate ongoing data updates—for example, rolling forecasts—you can run Process Plan Files to bring updated data from the database into plan files when that data is available. This may be necessary if your end users do not have the ability to run Axiom queries in plan files.

Process Plan Files is only available to administrators and to users with the **Process Plan Files** security permission. Users with this permission can process any plan file where they have at least read-only access, including the ability to run Axiom queries and save data to the database as part of the process. However, the user must have read/write access to the file in order to save the file as part of the process.

**NOTE:** This topic discusses how to run Process Plan Files interactively, from the file group menu. Alternatively, you can run Process Plan Files using Scheduler, and schedule it for automated execution.

**To process plan files:**

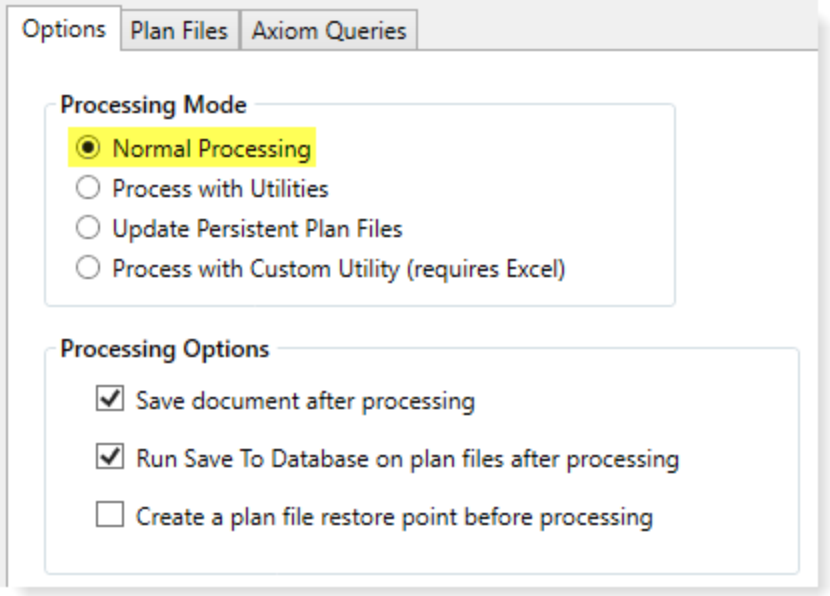
1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Process Plan Files**.



**TIP:** You can also access Process Plan Files in the Explorer task pane.

The **Process Plan Files** dialog opens.

2. On the **Options** tab, verify that **Normal Processing** is selected as the **Processing Mode**. This is the standard processing option that is used to refresh data in plan files and save data.




The other processing options apply to special use cases. For more information on these options, see the following:

- Processing plan files with utilities
- Updating persistent plan files
- Processing plan files with a custom utility

3. On the **Options** tab, complete the **Processing Options** as needed.

| Option | Description |
|---|---|
| Save document after processing | Specifies whether plan files are saved during processing. This option is selected by default.<br><br>This option does *not* cause a save-to-database to be performed—that option must be selected separately.<br><br>**NOTES:**<br>• If this option is not selected, then the utility will open the file as read-only and will not attempt to acquire the document lock before processing.<br>• If the file group uses virtual plan files, this option does not apply because the plan files cannot be saved. However, if the option is enabled, Axiom will attempt to acquire the document lock before processing, which is not necessary. This option should not be enabled when processing virtual plan files. |
| Run Save To Database on plan files after processing | Specifies whether a save-to-database is performed in plan files during processing. This option is selected by default.<br><br>This option does *not* cause the file itself to be saved—that option must be selected separately. It is not required to save the file in order to perform a save-to-database. |
| Create a plan file restore point before processing | If selected, then a plan file restore point will be created before processing begins. This option is not selected by default.<br><br>Restore points can be used to restore plan files to the state they were in before changes were made. For more information, see Restoring plan files from restore points.<br><br>**NOTE:** If the file group uses virtual plan files, this option does not apply. Plan files are not saved and therefore restore points are irrelevant. |

4. On the **Options** tab, complete the remaining options as needed.

   By default, plan files are processed on the Axiom Application Server, but you can optionally process plan files locally if desired (ideally, only when processing a small handful of plan files). For more information on these processing options, see Options tab.

5. On the **Plan Files** tab, specify the plan files to process.

   You can process all plan files that you have access to, or process only a subset of plan files. When processing a subset, you can manually select individual files to process, or you can define a filter to process the plan files that meet the filter. For more information on selecting plan files to process, see Plan Files tab.

6. On the **Axiom queries** tab, specify the Axiom queries to run in plan files during processing.

Only the selected Axiom queries will be run in plan files during processing. If no queries are selected, then no queries will be run, even if the query is configured to run on open. The "refresh on open" option for Axiom queries is ignored during processing.

Axiom queries are listed per template, but will be evaluated separately per plan file. For more information on selecting Axiom queries to run, see Axiom Queries tab.

7. Click **OK** to begin the process. The following prompts display before processing begins:
    - If your selections include any plan codes that do not yet have a plan file, then you are first prompted to decide whether you want to create plan files for these codes. If you click **Yes**, then Axiom attempts to create the plan files before proceeding with processing. You must have the appropriate security permissions to create these plan files. If you click **No**, then Axiom excludes those codes from the list of plan files to process (as shown in the next prompt).
    - A message box displays the number of plan files that you are about to process, and prompts you to confirm that you want to continue. Click **Yes** to continue with the processing. If the number of plan files is not as expected, then you can click **No** to cancel the process and return to the **Process Plan Files** dialog.

If you chose to process plan files on the Axiom server rather than the local client, then a Scheduler job is automatically placed in the queue for processing and you will be notified of the job results when it is complete. You can work on other Axiom tasks while the job is processed.

If you chose to process plan files on your local client, then a status bar displays as files are processed. When the process is complete, a dialog opens to display the results.

## How plan files are processed

When plan files are processed using Process Plan Files in Normal Processing mode, the following occurs for each plan file:

1. The plan file is opened and calculated. Data lookups that are configured to refresh on open will be run, but Axiom queries are not. The "refresh on open" setting is ignored for Axiom queries when using Process Plan Files.
2. Axiom queries are run in the plan file as follows:
    - If an Axiom query was selected in the Process Plan Files dialog, and the plan file was built using that template, then that query is eligible to be run in the plan file.
    - Before running the query, Axiom first checks the query settings in the plan file itself. If the query is active and **Refresh during document processing** is enabled for the query, then it will be run. If either of those settings are disabled, then the query will not be run. (This is how a selected query may be executed in some plan files but not others, depending on the query settings in each plan file.)
3. If **Run Save To Database on plan files after processing** was selected in the Process Plan Files dialog, then a save-to-database is executed in the plan file after Axiom queries are run.

4. If **Save document after processing** was selected in the Process Plan Files dialog, then the plan file is saved.

## Processing plan files with utilities

You can use the **Process Plan Files** feature to iteratively process a set of utilities per plan file, so that the utilities are processed using data for each plan file. This feature is primarily intended to support form-enabled plan files that use embedded forms, where the content of each plan file is sourced from various utility files instead of defined within the plan file itself. For more information on when utility processing is needed, see Using utility processing for plan files.

When a plan file is processed with utilities, the plan file itself is not processed. The plan file is opened and refreshed to initially set shared variables and to obtain the list of utilities to process for that plan file. Then, each utility is opened, refreshed, and a save-to-database is executed. Shared variables are automatically passed to each utility so that data queries can be filtered by plan code. Document variables can optionally be passed to each utility as well.

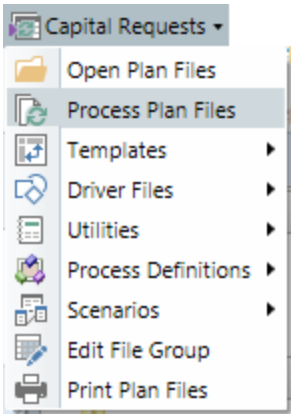
In order to process with utilities, the template must contain a ProcessPlanFileUtilities data source. For more information on creating this data source, see Creating ProcessPlanFileUtilities data sources.

Process Plan Files is only available to administrators and to users with the **Process Plan Files** security permission. Users with this permission can process any plan file where they have at least read-only access. The user executing the processing does *not* need to have security access to the utilities being processed.

**NOTE:** This topic discusses how to run Process Plan Files interactively, from the file group menu. Alternatively, you can run Process Plan Files using Scheduler, and schedule it for automated execution.

**To process plan files with utilities:**

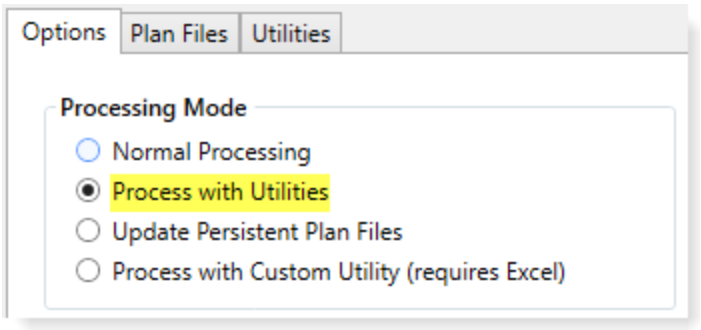
1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Process Plan Files**.



**TIP:** You can also access Process Plan Files in the Explorer task pane.

The **Process Plan Files** dialog opens.

2. On the **Options** tab, select **Process with Utilities** if it is not already selected.




If the file group is configured so that utility processing is the default processing mode, then this mode is selected by default. This is controlled by the **Process Plan Files with Utilities** option in the file group properties.

3. On the **Options** tab, complete the other processing options as needed.

By default, plan files are processed on the Axiom Application Server, but you can optionally process plan files locally if desired (ideally, only when processing a small handful of plan files). For more information on these processing options, see Options tab.

4. On the **Plan Files** tab, specify the plan files to process.

You can process all plan files that you have access to, or process only a subset of plan files. When processing a subset, you can manually select individual files to process, or you can define a filter to process the plan files that meet the filter. For more information on selecting plan files to process, see Plan Files tab.

5. On the **Utilities** tab, specify the data source to use for processing. This data source defines the list of utilities to be processed per plan file.

Data sources are listed per template, but will be evaluated separately for each plan file. For more information on selecting the data source for processing, see Utilities tab.

6. Click **OK** to begin processing.

A message box displays the number of plan files you are about to process, and asks you to confirm that you want to continue. Click **Yes** to continue. If the number of plan files is not as expected, you can click **No** to cancel the process and return to the **Process Plan Files** dialog.

If you chose to process plan files on the Axiom server rather than the local client, then a Scheduler job is created and you will be notified of the job results when it is complete. You can work on other Axiom tasks while the job is processed.

If you chose to process plan files on your local client, then a status bar displays as files are processed. When the process is complete, a dialog opens to display the results.

### How plan files are processed

When plan files are processed using Process Plan Files in Process with Utilities mode, the following occurs for each plan file:

1. The plan file is opened and refreshed. This includes running data lookups configured to refresh on open, and running eligible Axiom queries. The following data is read from the plan file:
    - The initial values for shared variables are set, so that they can be passed to the utilities.
    - The list of utilities to process is read from the ProcessPlanFileUtilities data source in the plan file. Only utilities that are flagged as enabled in the data source will be processed for this plan file. If the plan file contains multiple data sources, the data source that was selected in the Process Plan Files dialog is used.
2. Each utility is opened and processed as follows, starting with the first utility in the list and continuing downward:
    - The current set of shared variables is passed to the utility. Document variables, if defined in the data source for the utility, are also passed.
    - The utility is refreshed, including running data lookups configured to refresh on open, and running eligible Axiom queries.
    - A save-to-database is executed in the utility.
    - Shared variables are updated for any changes made in the utility. These variables are then passed to the next utility to be processed. If one utility depends on a value set in another utility, then these utilities must be processed in the appropriate order.

Active Axiom queries are eligible for processing in the plan file and in the utilities if **Refresh During Document Processing** is enabled. **Refresh on Open** queries are also run in the utilities only.

## Updating persistent plan files after creation

If a file group uses persistent plan files (the default behavior), then once plan files are built out, the plan files are separate entities that are not updated for any further changes made to the template or to calc methods. This is one reason why it is very important to perform comprehensive testing of plan files before rolling them out to end users. However, sometimes clients discover a change or correction that needs to be made to the template design or to a calc method after plan files have been built.

If this situation occurs, there are two ways that you can address it:

- If it is possible to rebuild plan files, then you should make the change to the template and/or calc method, and then rebuild plan files. This is the best and easiest approach if it is feasible.

- However, in some cases, rebuilding plan files is not a viable option. Your end users may have already started inputting planning data into plan files, which means that rebuilding plan files would lose this data in the files. In that case, you can use the **Update Persistent Plan Files** option of **Process Plan Files** to make targeted changes to plan files in batch.

The Update Persistent Plan Files option provides a find-and-replace approach to updating plan files. You can search for certain locations, text, or formulas in the plan files, and then update them with new text, formulas, or formatting.

In order to use Update Persistent Plan Files, you must first create a report file with a special control sheet that configures the update actions. Once you create this file, you can then go to the Process Plan Files utility, select the Update Persistent Plan Files option, and select the report file that you have configured for the update. When plan files are processed, the settings in the report file determine what actions are performed on the plan files.

**IMPORTANT:** Update Persistent Plan Files is an advanced feature. Please contact Axiom Support if you need assistance in configuring the utility or determining the best approach to addressing the issue you want to fix in your plan files.

**NOTES:**

- If the update that you need to make is in a calc method only, you may be able to use the **Apply Calc Method Changes** utility instead. Using this utility, you can update all or part of a given calc method in plan files. For more information, see Updating plan files for calc method changes .
- The need to update plan files after creation does not apply if the file group uses virtual plan files. In this case, you can simply update the template and/or calc methods as needed, and the changes will be reflected in the plan files the next time that they are accessed. This is because virtual plan files are re-created from template each time they are accessed. If the change affects the calculation of planning data, then after making changes you may want to run Process Plan Files with normal processing (or with utility processing if appropriate) in order to save updated data to the database.

## Creating the report file with the plan file update configuration

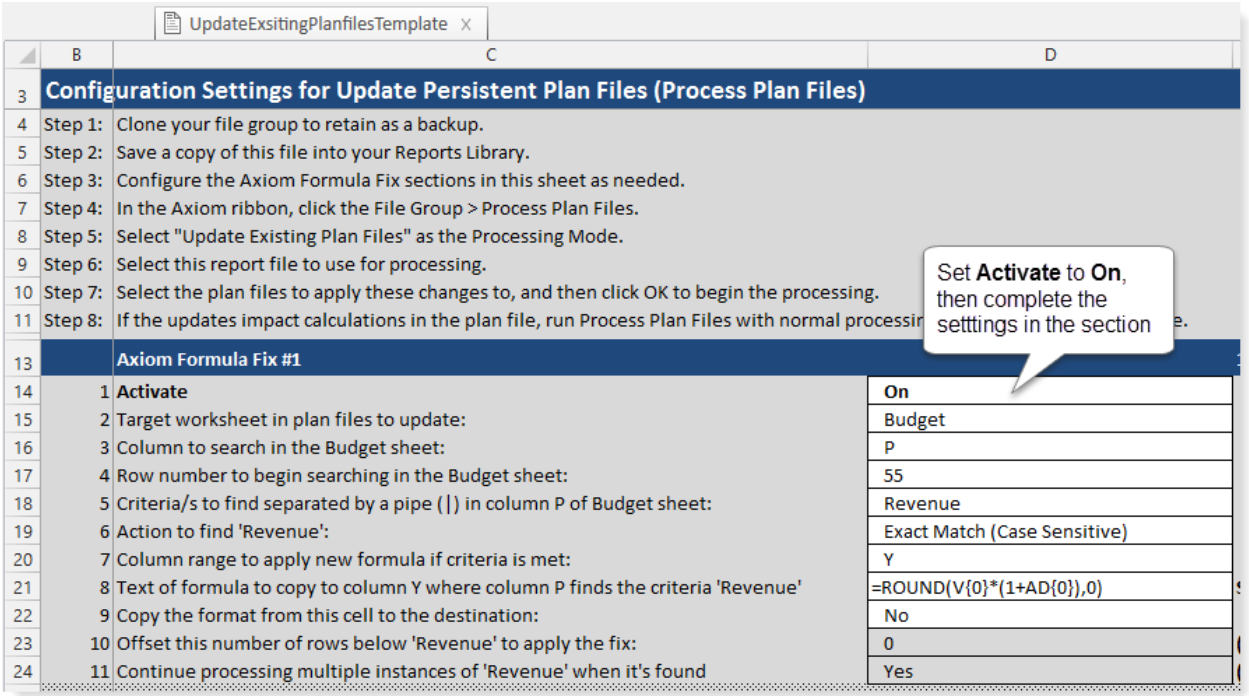
In order to use Update Persistent Plan Files, you must create a report file with a **PlanFileReconfig_ ControlSheet**, and then configure the settings in this control sheet.

Currently, the only way to create a report file with this control sheet is to save a copy of the following template file to the Reports Library:

```
\Axiom\Axiom System\Document Templates\Support
Utilities\UpdateExistingPlanfilesTemplate.xlsx
```

Only administrators can access files in the Axiom System section of Axiom Explorer, so the first time you need to use Update Persistent Plan Files, an administrator must save a copy of this file. Later, any other user with access to the copied file and the ability to save to a report folder can create a new version of the file for use with the Update Persistent Plan Files Utility.

Once the report file is created, use the PlanFileReconfig_ControlSheet to configure the update operation. There are 5 **Axiom Formula Fix** sections that you can use to define find-and-replace style operations. To activate a section for processing, set **Activate** to **On**, then complete the settings in that section.




For more information on these settings and example configurations, see Configuration settings for Update Persistent Plan Files.

**IMPORTANT:** The configuration settings in the PlanFileReconfig_ControlSheet should be based on actual changes that you have made to the template and/or calc methods. Even though the existing plan files will not be affected by changes to the template or calc methods, you should go through the process of updating these items to help ensure that the changes are accurate and to ensure that any plan files created in the future have the desired updates.

## Executing Process Plan Files using Update Persistent Plan Files

After you have created the configuration file for the update operation (see previous section), you can run Process Plan Files using the Update Persistent Plan Files processing mode.

Before running this utility for all plan files that you need to update, it is recommended to run it on a single test plan file first, to see if plan files are being updated as you expect. Once you have verified the operation on a test file, you can then run the utility on all plan files.

When you run Update Persistent Plan Files, a plan file restore point is automatically created before the update process is executed. If needed, you can use that restore point to restore plan files to the state they were in before running the utility. For more information, see Restoring plan files from restore points. If you want extra backup, you can optionally clone the file group (including plan files) before running the utility.

Process Plan Files is only available to administrators and to users with the **Process Plan Files** security permission.

**NOTE:** This topic discusses how to run Process Plan Files interactively, from the file group menu. Alternatively, you can run Process Plan Files using Scheduler, and schedule it for automated execution.

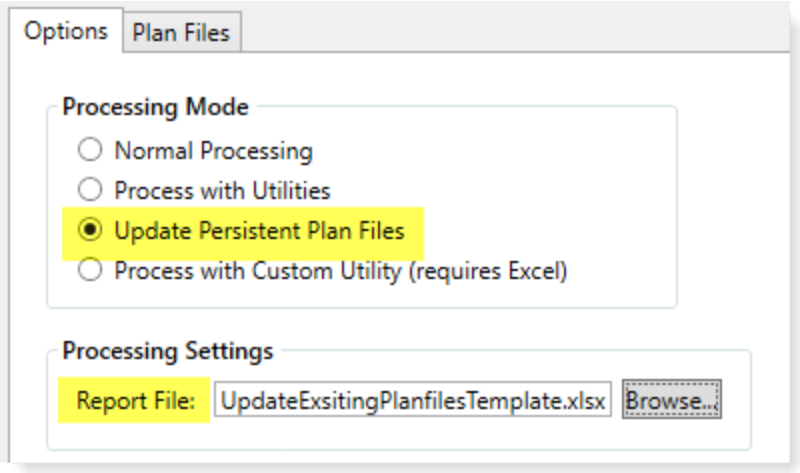
**To update persistent plan files using Process Plan Files:**

1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Process Plan Files**.

   **TIP:** You can also access Process Plan Files in the Explorer task pane.

   The **Process Plan Files** dialog opens.

2. On the **Options** tab, for the Processing Mode, select **Update Persistent Plan Files**.




   The dialog updates to show the relevant options for this processing mode.

3. On the **Options** tab, for **Report File**, specify the file that contains the configuration settings for processing. Click the **Browse** button to select the report file.

The file must be saved to the Reports Library and must contain a PlanFileReconfig_ControlSheet. See the previous section for more information on creating the file and completing the configuration settings.

4. On the **Options** tab, complete the remaining processing options as needed. For more information on these processing options, see Options tab.
5. On the **Plan Files** tab, specify the plan files to process.

   You can process all plan files that you have access to, or process only a subset of plan files. When processing a subset, you can manually select individual files to process, or you can define a filter to process the plan files that meet the filter. For more information on selecting plan files to process, see Plan Files tab.

   **IMPORTANT:** Be sure to test the update configuration on a single representative plan file before processing all plan files that need to be updated. If the update does not work as expected, you can adjust the configuration settings in the report file, and test again.

6. Click **OK** to begin processing.

   A message box displays the number of plan files you are about to process, and asks you to confirm that you want to continue. The message also informs you that a restore point will be created before the process begins, so that if necessary you can restore any plan files to the state they were in before running the utility. Click **Yes** to continue. If the number of plan files is not as expected, you can click **No** to cancel the process and return to the **Process Plan Files** dialog.

If you chose to process plan files on the Axiom server rather than the local client, then a Scheduler job is automatically placed in the queue for processing and you will be notified of the job results when it is complete. You can work on other Axiom tasks while the job is processed.

If you chose to process plan files on your local machine, then a status bar displays as files are processed. When the process is complete, a dialog opens to display the results.

### After processing

The Update Persistent Plan Files option simply saves the updated plan files; it does not save data to the database. If the changes made to plan files impact the calculation of planning data, then you should run **Normal Processing** on the updated plan files to save the changed data to the database.

## Configuration settings for Update Persistent Plan Files

When using the **Update Persistent Plan Files** processing mode for Process Plan Files, the settings for the update operation are defined in a report file with a special control sheet of **PlanFileReconfig_ ControlSheet**. This topic is a reference for the settings in this control sheet, and also contains some configuration examples.

## Axiom Formula Fix settings

The PlanFileReconfig_ControlSheet has 5 Axiom Formula Fix sections that can be used to define search-and-replace operations for existing plan files. To use a section, set Activate to On and then complete the settings in that section.

| Item | Description |
|---|---|
| Activate | Specifies whether the section is active. If **On**, the settings in this section will be applied when Update Persistent Plan Files is run. If **Off**, the settings in this section are ignored. |
| Target worksheet in plan files to update | The name of the sheet in the plan files to update. Does not apply when the action is **Simple Find/Replace Entire Workbook**. |
| Column to search in the target sheet | The column that contains the search criteria. Only applies when the action is **Exact Match**, **Partial Match Value**, or **Partial Match Formula**.<br><br>This can be a different column than where the fix needs to be applied. For example, you may need to search for a word, value, or formula in column P to find the correct row, but the actual update is applied to a different column or columns in the same row, such as AI:AT. If the update location is not in the same row as the search criteria, you can use the offset setting. |
| Row number to begin searching in the target sheet | The row in which to begin searching, within the specified search column. Only applies when the action is **Exact Match**, **Partial Match Value**, or **Partial Match Formula**, and is optional in these cases.<br><br>This setting allows you to skip the "setup" areas at the top of the sheet, if appropriate. |
| Criteria to find separated by a pipe in search column of target sheet | The text, value, or formula to locate in the target column. Does not apply when the action is **Specific Address**.<br><br>You can specify multiple search criteria, separated by the pipe character. If multiple search criteria are specified, they are treated as an OR statement (meaning, the cell only has to match one of the criteria). |

| Item | Description |
| --- | --- |
| Action to find search criteria | Specifies how the search finds matches. Select one of the following:<br>• **Exact Match (Case Sensitive)**: Searches for an exact text match, then makes the change in the target row.<br>• **Partial Match Value**: Searches for a partial match on values only, then makes the change on the target row.<br>• **Partial Match Formula**: Searches for a partial match on formulas only, then makes the change on the target row.<br>• **Simple Find / Replace Target Worksheet**: Replaces all instances of the search criteria in the target worksheet with the designated fix. Ignores the search column, search row, and column range settings. Formats are not copied.<br>• **Simple Find / Replace Entire Workbook**: Replaces all instances of the search criteria with the designated fix. Ignores the target worksheet, search column, search row, and column range settings. Formats are not copied.<br>• **Specific Address**: Replaces the contents of the specified cell with the designated fix. Ignores the search criteria, target worksheet, search column, search row, and column range settings. |
| Column range to apply new formula if criteria is met | Specifies the column range in which to apply the update if the search criteria is found. Only applies when the action is **Exact Match**, **Partial Match Value**, or **Partial Match Formula**.<br>For example, specify P:P to make the change in column P only, or specify P:Z to make the change in a range of columns.<br>By default, the update is applied in the row where the search criteria is found, unless a row offset is specified in the **Offset this number of rows** field. |
| Enter the specific cell reference | Specifies the cell to update when using the **Specific Address** action. Enter the cell reference in the format A1. The sheet for the cell reference is the specified target worksheet.<br>**NOTE:** This is the same field as the **Column range to apply new formula if criteria is met** field. The text of the field changes when Specific Address is the selected action. |

| Item | Description |
| --- | --- |
| Text of formula to copy | Defines the text, value, or formula to copy to the update destination in plan files. Only the cell contents are copied, not the formatting. If you want to copy formatting as well (or by itself), then use the **Copy the format from this cell to the destination** field.<br><br>If you want to copy a formula to plan files, this cell must contain the formula string formatted as text. By default, the cell is formatted as Text so that the formula will not resolve. For example, the contents of the cell should look something like the following, instead of a resolved formula:<br><br>`=ROUND(V{0}*(1+AD{0}),0)`<br><br>To copy a formula from another spreadsheet (such as the revised template), you can place your cursor in the cell, highlight the contents of the formula bar, and then copy and paste the formula string to this cell.<br><br>Any variable references to rows in the formula should be replaced with a row offset in curly brackets. For example, D{0} is the current row, D{-4} is four rows above the current row, and D{4} is four rows below the current row. |
| Copy the format from this cell to the destination | Specifies whether formatting is copied to the update destination. By default, this is set to **No**, which means that the existing formatting of the update destination is preserved.<br><br>If you want to change formatting in plan files, select one of the following and then format this cell as desired:<br>• Select **Yes** if you want to copy formatting to the update destination in addition to copying the text, value, or formula.<br>• Select **Only Format - No Formula Change** if you want to copy *only* formatting to the update destination, leaving the existing cell contents intact.<br><br>This option does not apply if either of the "simple find / replace" actions are used.<br><br>**IMPORTANT:** The formatting is copied from this cell, *not* from the **Text of formula to copy** cell. If you are updating cell contents and format, then the contents go in the "text of formula" cell and the formatting goes in the "copy the format" cell. |

| Item | Description |
|---|---|
| Offset this number of rows below the search criteria to apply the fix | Optional. Specify the number of rows below the search criteria in which to apply the update. For example, you might search for the text Statistics and want to apply the update to a column location that is two rows down instead of in the current row.<br><br>If the update location is above the search criteria, you can enter a negative number. An entry of 2 applies the fix 2 rows below the search criteria, whereas an entry of -2 applies the fix 2 rows above the search criteria.<br><br>Only applies when the action is **Exact Match**, **Partial Match Value**, or **Partial Match Formula**. |
| Continue processing multiple instances of the search criteria when it's found (advanced) | Specifies whether the update process continues after the first match or stops. By default, this is set to Yes, which means the update process looks for all eligible matches and updates each one. If set to No, then the update process stops after updating the first match.<br><br>Does not apply when using either of the **Simple Find/Replace** options, or when using **Specific Address**. |

## Example 1: Exact Match

The Exact Match option finds the search criteria in the specified column and applies the update to the specified location. It searches values for exact matches and is case-sensitive.

The following example searches for the text "Revenue" in column S, and then replaces the formula in column AA on the same row. The formula uses the variable {0} for the current row number:

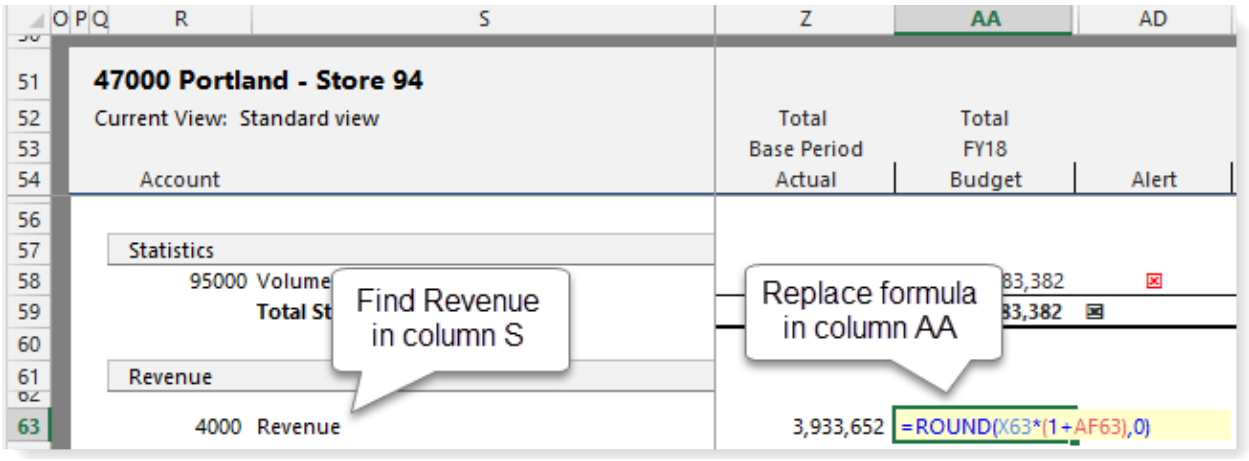
```
=ROUND(X{0}*(1+AF{0}),0)
```

Where {0} is replaced with the current row number when the update occurs.

| | B | C | D |
|---|---|---|---|
| 3 | **Configuration Settings for Update Persistent Plan Files (Process Plan Files)** | | |
| 13 | | **Axiom Formula Fix #1** | |
| 14 | 1 | **Activate** | **On** |
| 15 | 2 | Target worksheet in plan files to update: | Budget |
| 16 | 3 | Column to search in the Budget sheet: | S |
| 17 | 4 | Row number to begin searching in the Budget sheet: | 55 |
| 18 | 5 | Criteria/s to find separated by a pipe (\|) in column S of Budget sheet: | Revenue |
| 19 | 6 | Action to find 'Revenue': | Exact Match (Case Sensitive) |
| 20 | 7 | Column range to apply new formula if criteria is met: | AA |
| 21 | 8 | Text of formula to copy to column AA where column S finds the criteria 'Revenue' | =ROUND(X{0}*(1+AF{0}),0) |
| 22 | 9 | Copy the format from this cell to the destination: | No |
| 23 | 10 | Offset this number of rows below 'Revenue' to apply the fix: | 0 |
| 24 | 11 | Continue processing multiple instances of 'Revenue' when it's found | Yes |

*Example Exact Match configuration*

The following example shows how the update is applied in a plan file:




*Example updated plan file*

## Example 2: Partial Match Value

The Partial Match Value option finds the search criteria in the specified column and applies the update to the specified location. It searches values for partial matches.

The following example searches for the text "Research" in column S, and then replaces the formulas in columns AK:AV on the same row. The formula uses the variable {0} for the current row number:

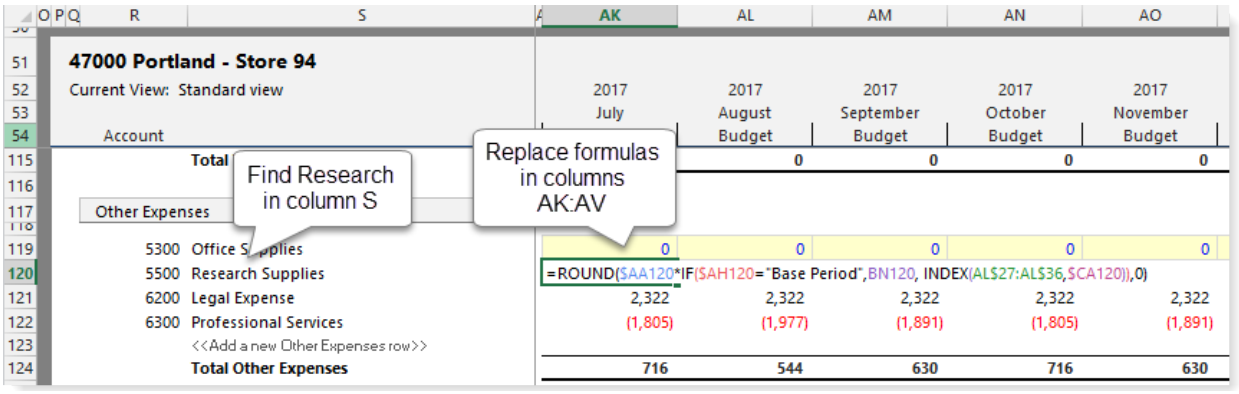
```
=ROUND($AA{0}*IF($AH{0}="Base Period",BN{0}, INDEX(AL$27:AL$36,$CA{0})),0)
```

Where {0} is replaced with the current row number when the update occurs.

| | B | C | D |
|---|---|---|---|
| 3 | **Configuration Settings for Update Persistent Plan Files (Process Plan Files)** | | |
| 26 | | **Axiom Formula Fix #2** | |
| 27 | 1 | **Activate** | **On** |
| 28 | 2 | Target worksheet in plan files to update: | Budget |
| 29 | 3 | Column to search in the Budget sheet: | S |
| 30 | 4 | Row number to begin searching in the Budget sheet: | 55 |
| 31 | 5 | Criteria/s to find separated by a pipe (\|) in column S of Budget sheet: | Research |
| 32 | 6 | Action to find 'Research': | Partial Match Value |
| 33 | 7 | Column range to apply new formula if criteria is met: | AK:AV |
| 34 | 8 | Text of formula to copy to column AK:AV where column S finds the criteria 'Research' | =ROUND($AA{0}*IF($AH{0}="Base Period",BN{0}, INDEX(AL$27:AL$36,$CA{0})),0) |
| 35 | 9 | Copy the format from this cell to the destination: | |
| 36 | 10 | Offset this number of rows below 'Research' to apply the fix: | 0 |
| 37 | 11 | Continue processing multiple instances of 'Research' when it's found | Yes |

*Example Partial Match Value configuration*

The following example shows how the update is applied in a plan file:




*Example updated plan file*

## Example 3: Partial Match Formula

The Partial Match Formula option finds the search criteria in the specified column and applies the update to the specified location. It searches within formulas for partial matches.

The following example searches for the partial string "=ROUND(X" within formulas in column AA, and then replaces those formulas in the same column and row (search column and update column range are the same column). The formula uses the variable {0} for the current row number:

```
=ROUND(X{0}*(1+AF{0}),0)
```

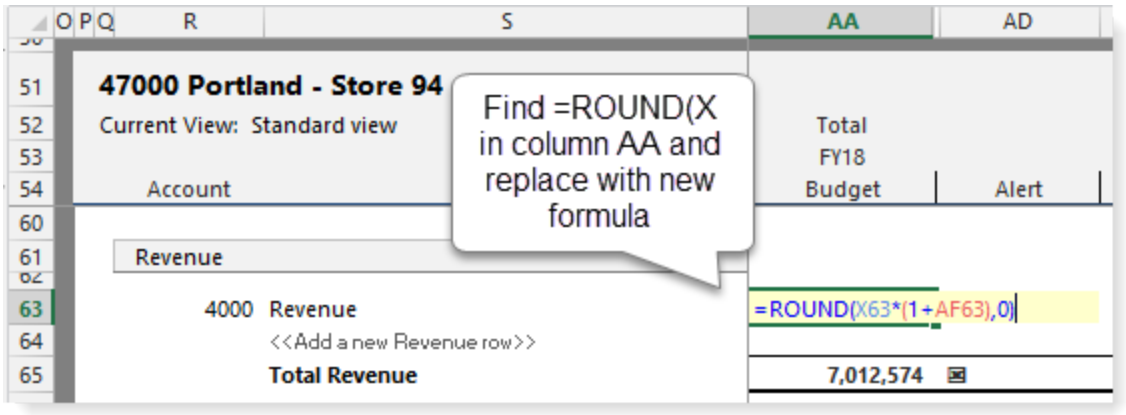
Where {0} is replaced with the current row number when the update occurs.

Note that you need to change the number format of the search criteria cell to Text so that you can enter the partial formula string as text (otherwise it will resolve as an invalid formula).

| | B | C | D |
|---|---|---|---|
| 3 | **Configuration Settings for Update Persistent Plan Files (Process Plan Files)** | | |
| 39 | | **Axiom Formula Fix #3** | |
| 40 | 1 | **Activate** | **On** |
| 41 | 2 | Target worksheet in plan files to update: | Budget |
| 42 | 3 | Column to search in the Budget sheet: | AA |
| 43 | 4 | Row number to begin searching in the Budget sheet: | 55 |
| 44 | 5 | Criteria/s to find separated by a pipe (\|) in column AA of Budget sheet: | =ROUND(X |
| 45 | 6 | Action to find '=ROUND(X': | Partial Match Formula |
| 46 | 7 | Column range to apply new formula if criteria is met: | AA |
| 47 | 8 | Text of formula to copy to column AA where column AA finds the criteria '=ROUND(X' | =ROUND(X{0}*(1+AF{0}),0) |
| 48 | 9 | Copy the format from this cell to the destination: | No |
| 49 | 10 | Offset this number of rows below '=ROUND(X' to apply the fix: | 0 |
| 50 | 11 | Continue processing multiple instances of '=ROUND(X' when it's found | Yes |

*Example Partial Match Formula configuration*

The following example shows how the update is applied in a plan file:




*Example updated plan file*

## Example 4: Simple Find/Replace

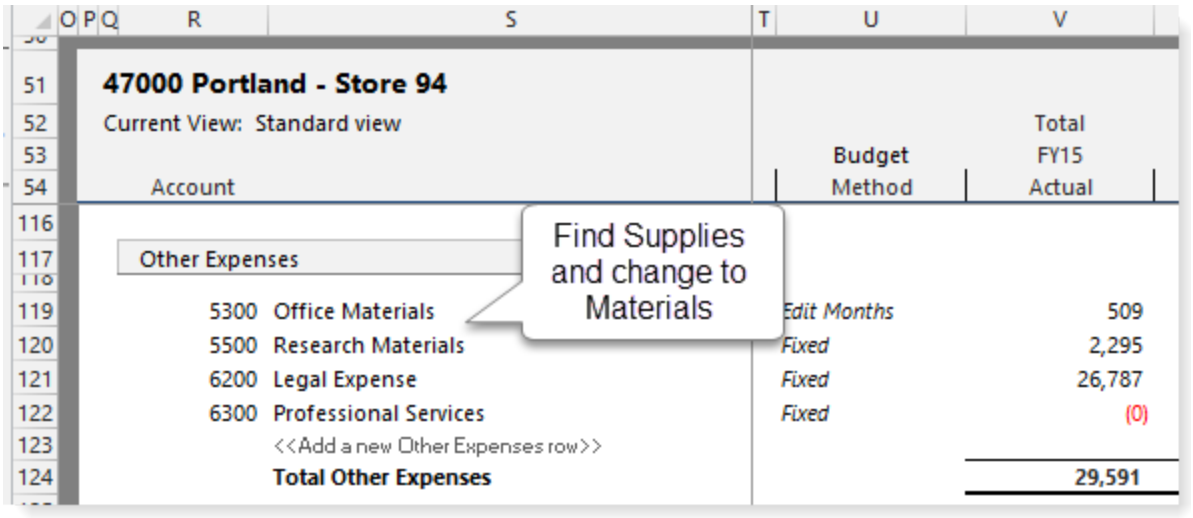
The Simple Find/Replace options find all instances of the search criteria and replace it. These options search within values for partial matches. The Simple Find/Replace Target Worksheet option only updates the specified target worksheet, whereas the Simple Find/Replace Entire Workbook option updates the entire workbook.

The following example searches for the text "Supplies" within the target worksheet, and replaces it with the text "Materials". Note that the search column, search row, and update column range fields are left blank since they do not apply in this case. If the action was Simple Find/Replace Entire Workbook instead, then the target worksheet would be blank as well.

| | B | C | D |
|---|---|---|---|
| 3 | **Configuration Settings for Update Persistent Plan Files (Process Plan Files)** | | |
| 52 | | **Axiom Formula Fix #4** | |
| 53 | 1 | **Activate** | **On** |
| 54 | 2 | Target worksheet in plan files to update: | Budget |
| 55 | 3 | Column to search in the Budget sheet: | |
| 56 | 4 | Row number to begin searching in the Budget sheet: | |
| 57 | 5 | Criteria/s to find separated by a pipe (\|): | Supplies |
| 58 | 6 | Action to find 'Supplies': | Simple Find/Replace Target Worksheet |
| 59 | 7 | Column range to apply new formula if criteria is met: | |
| 60 | 8 | Text of formula to copy to column where column finds the criteria 'Supplies' | Materials |
| 61 | 9 | Copy the format from this cell to the destination: | |
| 62 | 10 | Offset this number of rows below 'Supplies' to apply the fix: | 0 |
| 63 | 11 | Continue processing multiple instances of 'Supplies' when it's found | Yes |

*Example Simple Find/Replace configuration*

The following example shows how the update is applied in a plan file:




*Example updated plan file*

## Example 5: Specific Address

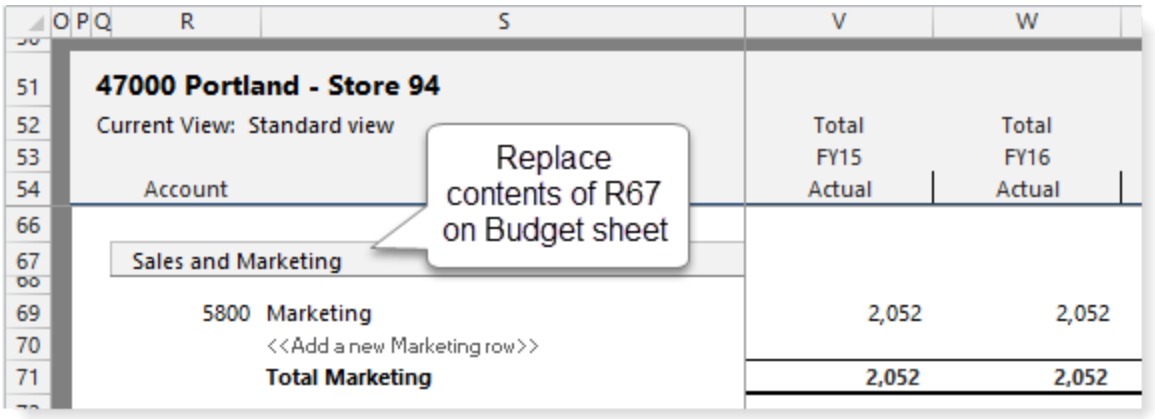
The Specific Address option finds the designated cell address and replaces the contents of it. This option is for cases where you know the specific location of the fix and it is the same location for all plan files that you need to update.

The following example finds cell R67 on the Budget sheet and replaces its contents with the text "Sales and Marketing". Note that the search column, search row, and search criteria fields are left blank since they do not apply in this case, and the column range field has been replaced by the cell address field.

| | B | C | D |
|---|---|---|---|
| 3 | **Configuration Settings for Update Persistent Plan Files (Process Plan Files)** | | |
| 65 | | **Axiom Formula Fix #5** | |
| 66 | 1 | **Activate** | **On** |
| 67 | 2 | Target worksheet in plan files to update: | Budget |
| 68 | 3 | Column to search in the Budget sheet: | |
| 69 | 4 | Row number to begin searching in the Budget sheet: | |
| 70 | 5 | Criteria/s to find separated by a pipe (|): | |
| 71 | 6 | Action to find: | Specific Address |
| 72 | 7 | Enter the specific cell address: | R67 |
| 73 | 8 | Text of formula to copy to column R67 where column finds the criteria " | Sales and Marketing |
| 74 | 9 | Copy the format from this cell to the destination: | |
| 75 | 10 | Offset this number of rows below " to apply the fix: | 0 |
| 76 | 11 | Continue processing multiple instances of " when it's found | Yes |

*Example Specific Address configuration*

The following example shows how the update is applied in a plan file:




*Example updated plan file*

## Processing plan files with a custom utility

You can use the Process Plan Files feature to run a custom utility on plan files. The custom utility must be a Microsoft Excel spreadsheet that contains a VBA module with the custom code. This file must be saved within the Axiom file system, in the Reports Library.

**IMPORTANT:** This feature should only be used to run "approved" custom utilities provided by an Axiom representative.

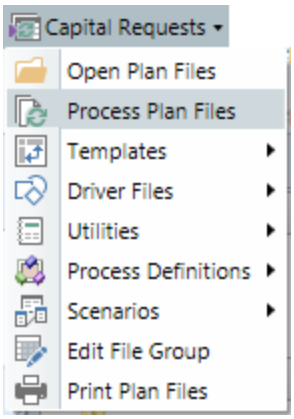
When you process plan files using a custom utility, you specify the utility to process, and the plan files to include in the processing. Processing can be performed on the local client or on the Axiom Scheduler server, however, the utility must be processed using Microsoft Excel. If your installation does not use Microsoft Excel on the Scheduler server, then you must process the utility on a client machine, running the Excel Client.

Process Plan Files is only available to administrators and to users with the **Process Plan Files** security permission.

**NOTE:** This topic discusses how to run Process Plan Files interactively, from the file group menu. Alternatively, you can run Process Plan Files using Scheduler, and schedule it for automated execution.

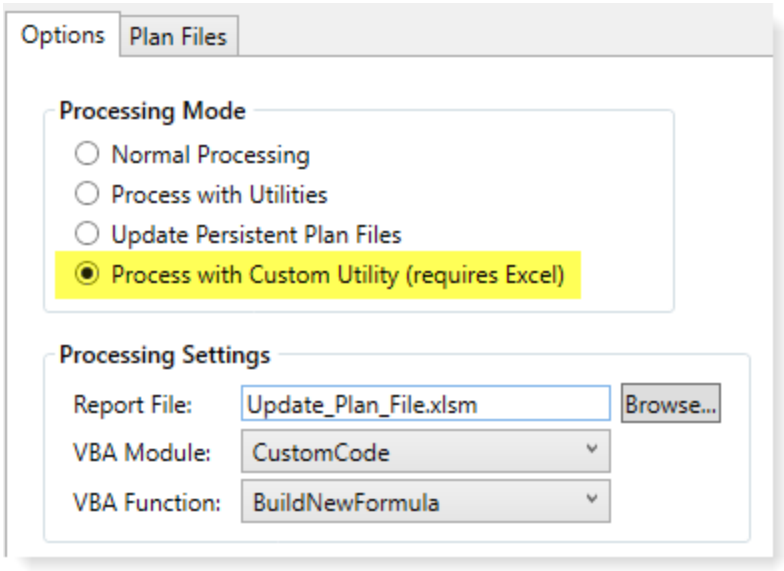
**To process plan files using a custom utility:**

1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Process Plan Files**.




The **Process Plan Files** dialog opens.

2. On the **Options** tab, for the Processing Mode, select **Process with Custom Utility**.




The dialog updates to show the relevant options for this processing mode.

3. On the **Options** tab, complete the **Processing Settings**:

| Item | Description |
|---|---|
| Report File | Click the **Browse** button to select the Microsoft Excel spreadsheet file that contains the VBA custom utility. The file must be saved in the Reports Library. |
| VBA Module | Select the VBA module to run as part of this utility. The drop-down list shows the VBA modules available in the selected file. |
| VBA Function | Select the VBA function to run as part of this utility. The drop-down list shows the VBA functions available in the selected module. |

4. On the **Options** tab, complete the remaining processing options as needed. For more information on these processing options, see Options tab.

   **NOTE:** Custom utilities can only be processed using Excel. If you have Excel on your Scheduler server, then you can process plan files on the **Axiom Server**. Otherwise, you must use the Excel Client and use **Local Axiom Client** processing.

5. On the **Plan Files** tab, specify the plan files to process.

   You can process all plan files that you have access to, or process only a subset of plan files. When processing a subset, you can manually select individual files to process, or you can define a filter to process the plan files that meet the filter. For more information on selecting plan files to process, see Plan Files tab.

6. Click **OK** to begin processing.

   A message box displays the number of plan files you are about to process, and asks you to confirm that you want to continue. The message also informs you that a restore point will be created before the process begins, so that if necessary you can restore any plan files to the state they were in before running the utility (see Restoring plan files from restore points). Click **Yes** to continue. If the number of plan files is not as expected, you can click **No** to cancel the process and return to the **Process Plan Files** dialog.

If you chose to process plan files on the Axiom server rather than the local client, then a Scheduler job is automatically placed in the queue for processing and you will be notified of the job results when it is complete. You can work on other Axiom tasks while the job is processed.

If you chose to process plan files on your local machine, then a status bar displays as files are processed. When the process is complete, a dialog opens to display the results.

### Notes for custom utility designers

The VBA function used in the Process Plan Files utility should return a string. Null or empty string values are interpreted as success. Any other string value is interpreted as an error, and the returned string is logged as the error message.

## Process Plan Files dialog

Use the Process Plan Files dialog to configure options for plan file processing. This topic provides a reference for the various options in this dialog. Process Plan Files is only available to administrators and to users with the **Process Plan Files** security permission.

The Process Plan Files dialog has several tabs to define different options. The tabs available in the dialog and the options on those tabs depend on the selected **Processing Mode** on the **Options** tab.

- **Options**: Defines the overall processing mode and processing options
- **Plan Files**: Specifies the plan files to process

- **Axiom Queries**: Specifies which Axiom queries to run in plan files (only applies to Normal Processing)
- **Utilities**: Specifies which data source to use for utility processing (only applies to Process with Utilities)

## Options tab

The Options tab defines various options for processing. Some options vary depending on the selected processing mode, while other options are available for all modes.

### General options

| Option | Description |
|---|---|
| Processing Mode | Select the type of processing to perform:<br>• **Normal Processing**: Plan files are opened, refreshed, and saved. You can configure which actions occur. For more information, see Processing plan files.<br>• **Process with Utilities**: A list of utilities is iteratively processed per plan file. Utilities are opened, refreshed with data for each plan code, and saved. This is primarily intended for processing form-enabled plan files that use embedded forms. For more information, see Processing plan files with utilities.<br>• **Update Persistent Plan Files**: Update existing plan files for text, formatting, or formula fixes. This is an advanced feature. For more information, see Updating persistent plan files after creation.<br>• **Process with Custom Utility**: Plan files are processed using a custom utility provided by Axiom Support. This is an advanced feature. For more information, see Processing plan files with a custom utility.<br>The default processing mode is Normal Processing. However, if the file group has been configured so that utility processing is the primary processing mode for that file group, then Process with Utilities is selected by default. |

| Option | Description |
| --- | --- |
| Process Plan Files on | Select one of the following:<br>• **Axiom Server** (default): Process the utility on an Axiom Scheduler server. This frees up local resources and allows you to work on other Axiom tasks while the utility is being run. When the utility is finished processing, you will receive an email with the results.<br>• **Local Axiom Client**: Process the utility on your local client machine. You cannot work on other Axiom tasks while the utility is being run, but when it is finished, the results display immediately.<br>You might want to process on the server if you are processing a large number of plan files and do not want to wait for the utility to complete before moving on to other tasks. However, if you are only processing a small number of plan files, and you want to see the results immediately, you can process on the client.<br>If you choose to process on the server, a Scheduler job will be created to process the plan files, with a start time of the current time. These requests are handled by the System.RefreshPlanFiles event handler in Scheduler. The job is run under the identity of the user who initiated the request.<br>**NOTES:**<br>• Process with Utilities mode can only be run in the Windows Client (or the Web Server engine). If you are currently in the Excel Client, the local client option is not available.<br>• Process with Custom Utility mode can only be run in Excel. If you are currently in the Windows Client, the local client option is not available. |
| Server Options | By default, **Process plan files using the Axiom Web engine** is selected and cannot be changed. The Web engine is the Excel-compatible spreadsheet engine used by the Axiom Windows Client and the Axiom Web Client to read Axiom files without using Excel.<br>In some older systems, the option may be editable, but you cannot disable the option if you are processing on the server. It is no longer supported to perform server-side processing using the Excel Client.<br>**NOTES:**<br>• This option is enabled by default and cannot be changed when using Process with Utilities mode.<br>• This option is disabled by default and cannot be changed when using Process with Custom Utility mode. |

| Option | Description |
|---|---|
| Notification Options | If the process is being executed on the Axiom Server, select one of the following options to specify how you want to be notified when the process is completed. The notification will include error information, if applicable.<br>• **Notification task pane** (default): Display the notification in the Notifications task pane.<br>• **Email notification**: Receive the notification by email, using your email address as defined in Axiom Security.<br>• **Both notifications**: Receive the notification by email and in the Notifications task pane. |

### Options for Normal Processing mode

If Normal Processing is the selected processing mode, the following additional processing options are available on the Options tab:

| Option | Description |
|---|---|
| Save document after processing | Specifies whether plan files are saved during processing. This option is selected by default.<br>This option does *not* cause a save-to-database to be performed—that option must be selected separately.<br>**NOTES:**<br>• If this option is not selected, then the utility will open the file as read-only and will not attempt to acquire the document lock before processing.<br>• If the file group uses virtual plan files, this option does not apply because the plan files cannot be saved. However, if the option is enabled, Axiom will attempt to acquire the document lock before processing, which is not necessary. This option should not be enabled when processing virtual plan files. |
| Run Save To Database on plan files after processing | Specifies whether a save-to-database is performed in plan files during processing. This option is selected by default.<br>This option does *not* cause the file itself to be saved—that option must be selected separately. It is not required to save the file in order to perform a save-to-database. |

| Option | Description |
| --- | --- |
| Create a plan file restore point before processing | If selected, then a plan file restore point will be created before processing begins. This option is not selected by default.<br><br>Restore points can be used to restore plan files to the state they were in before changes were made. For more information, see Restoring plan files from restore points.<br><br>**NOTE:** If the file group uses virtual plan files, this option does not apply. Plan files are not saved and therefore restore points are irrelevant. |

### Options for Process with Utilities

If **Process with Utilities** is the selected processing mode, there are no additional options on the Options tab.

Plan files are not saved when using Process with Utilities, and plan file restore points are not created. When using this mode, the processing is being performed in the utility files, not in the plan files, so it is not necessary to save the plan files. Additionally, in most cases the plan files used with this mode are virtual form-enabled plan files, so the save and restore options are irrelevant.

### Options for Update Persistent Plan Files

If **Update Persistent Plan Files** is the selected processing mode, the following additional option is available on the Options tab:

| Option | Description |
| --- | --- |
| Report File | Click the Browse button to select the report file that is configured with the PlanFileReconfig_ControlSheet. This file must be saved in the Reports Library.<br><br>This control sheet contains the settings that will be applied to plan files during processing. |

Plan files are always saved when using this processing option, and plan file restore points are always created before processing. A save-to-database is not performed in this mode, so if you need to save data, you should process plan files using Normal Processing after you have verified the results of the plan file update.

### Options for Process with Custom Utility

If **Process with Custom Utility** is the selected processing mode, the following additional options are available on the Options tab:

| Item | Description |
|---|---|
| Report File | Click the **Browse** button to select the Microsoft Excel spreadsheet file that contains the VBA custom utility. The file must be saved in the Reports Library. |
| VBA Module | Select the VBA module to run as part of this utility. The drop-down list shows the VBA modules available in the selected file. |
| VBA Function | Select the VBA function to run as part of this utility. The drop-down list shows the VBA functions available in the selected module. |

Plan files are always saved when using this processing option, and plan file restore points are always created before processing. A save-to-database is not performed in this mode, so if you need to save data, you should process plan files using Normal Processing after you have verified the results of the custom utility processing.

### Plan Files tab

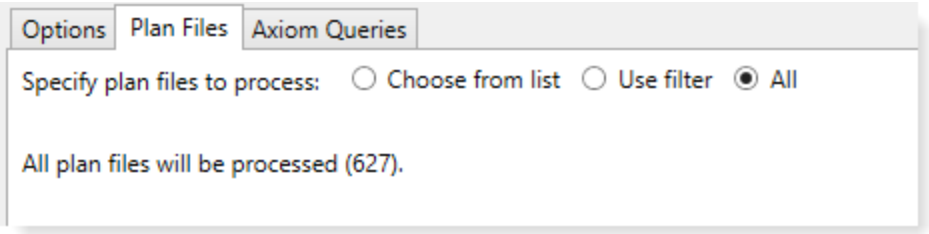
On the **Plan Files** tab, specify the plan files that you want to process. There are three different options that you can use to specify the plan files: **Choose from list**, **Use filter**, and **All**. You should use the option that corresponds to how many plan files you want to process—all plan files, or a subset of plan files. If you want to process a subset of plan files, you can select individual files to process or you can use a filter to define the subset.

**NOTES:**

- If a plan file is locked by another user, then it cannot be processed by the Process Plan Files utility. If you select a file that is currently locked by another user, then when you click **OK** to run the utility, a message box warns you of the locked file and prompts you to choose whether to continue. If you continue, the file lock will be checked again before the file is processed as part of the utility, and if the file is still locked then it will not be processed and an error will be noted in the results.
- If the file group uses a **Show on List** column, then any plan code that is set to **False** will not display in the plan file list and will be ignored when processing.

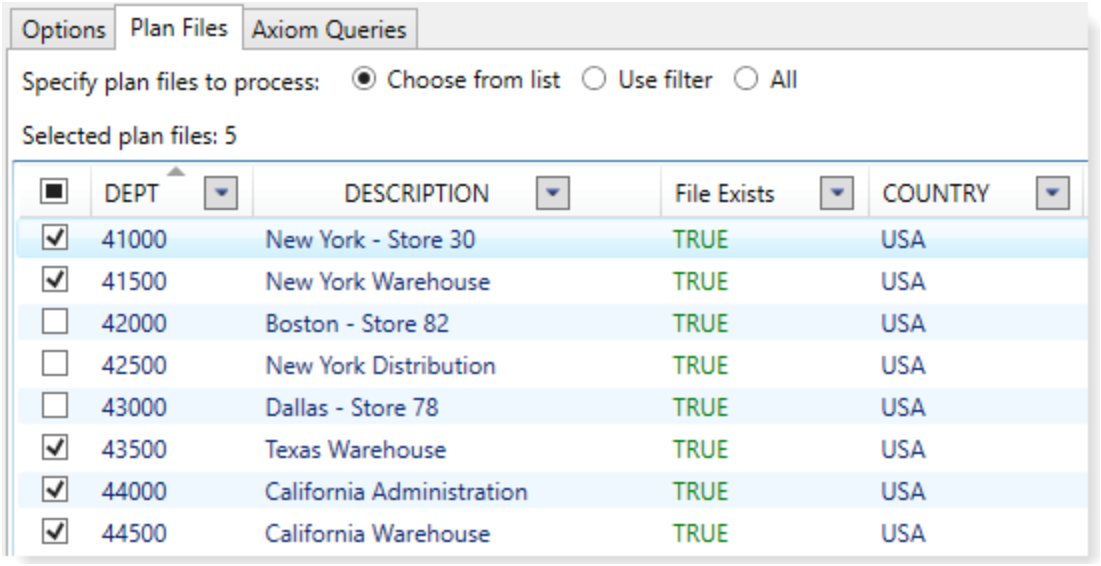
#### Process all plan files

To process all plan files, select **All**. When you start processing, Axiom will retrieve the full list of plan files that you have access to and process those plan files.



### Process selected plan files

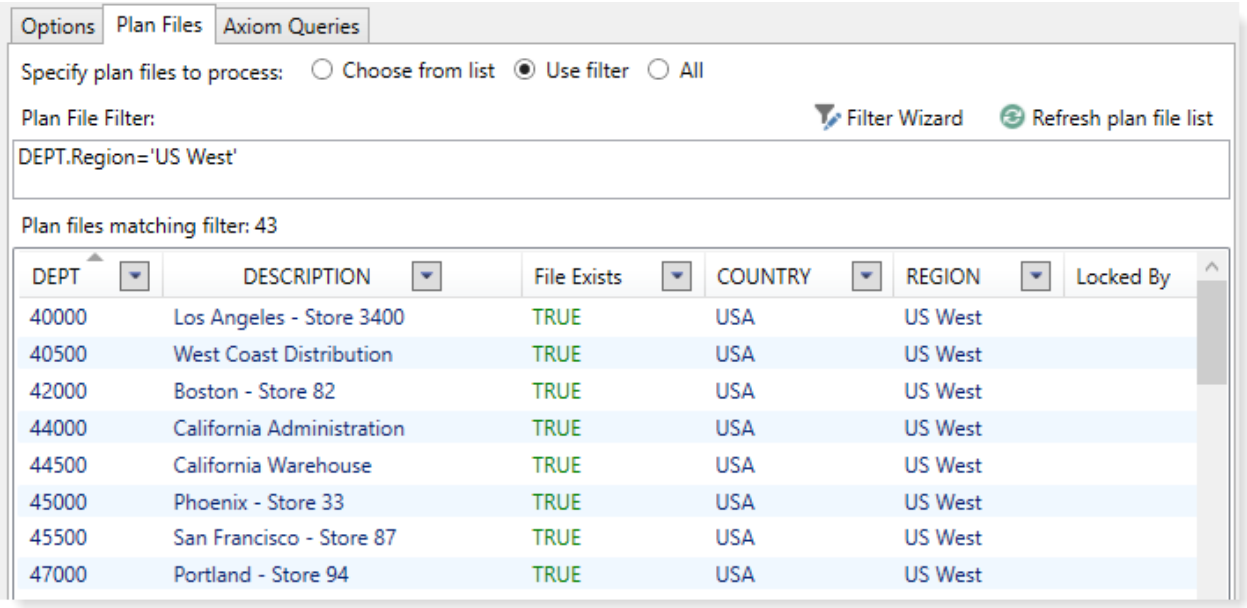
To process certain plan files, select **Choose from list**, and then select the check boxes for the plan files that you want to process. When you start processing, Axiom will process only the selected plan files.




To find the plan files you are looking for, you can sort, filter, and group the list using standard Axiom grid features. You can show additional columns and hide columns by right-clicking in the column header. If you have filtered the list, you can select the check box in the header to select only the plan files that currently display in the dialog.

### Process a filtered set of plan files

To use a filter to process a subset of plan files, select **Use Filter**. When you start processing, Axiom will process only the plan files that meet the filter.




You can use the Filter Wizard to create the filter, or you can manually type a filter criteria statement into the filter box. The filter must use the plan code table or a lookup table. For example: `DEPT.Region='US West'` where Dept is the plan code table.

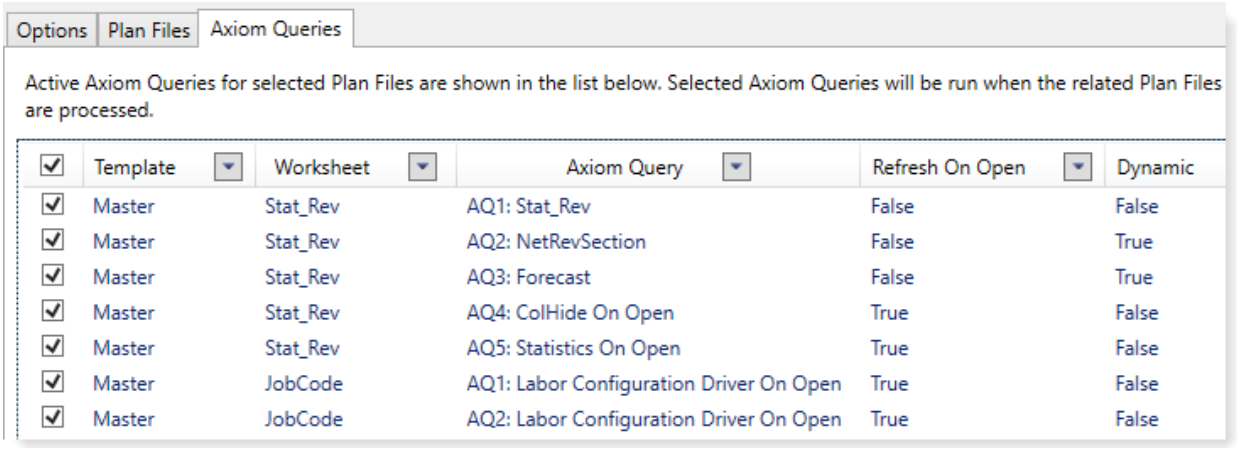
Once you have entered a filter, you can click **Refresh plan file list** to show the plan files that currently match the filter. The refresh feature is intended to help you determine whether you have defined the filter correctly.

**NOTE:** When you define a filter, all plan files that meet the filter are automatically processed. You do not need to select any plan files for processing. If instead you want to filter the list and then individually select some plan files, use the **Choose from list** option. You can use the grid features to filter the list, and then select the specific plan files that you want to process.

## Axiom Queries tab

On the **Axiom Queries** tab, select the queries that you want to run in the plan files. By default, all listed queries are selected. This tab only applies when using **Normal Processing** mode.

If you do not want to run a particular query, you can clear the check box. You can select or clear individual check boxes, or you can use the check box in the header to select or clear all queries currently displayed in the list. You can sort, filter, and group the list using standard Axiom grid functionality.




*Example Axiom Queries tab*

The list of Axiom queries is based on the source templates that were used to create the plan files. Only Axiom queries that meet the following criteria are eligible for selection:

- **Active** is set to **On**, or the setting uses a formula.
- **Refresh during document processing** is set to **On**.

If a query uses a formula for the Active setting, this means the query is dynamic and may or may not be run, depending on how the formula resolves in each plan file to be processed. When a particular plan file is processed, each selected query will be evaluated based on the current settings in that plan file. If both **Active** and **Refresh during document processing** are **On** for that plan file, then the query will be run. If either or both settings are **Off** for that plan file, the query will not be run. You can tell whether a query is dynamic or not by looking at the **Dynamic** column in the query list.

If a query is *not* selected on this tab, then that query will not be run in any plan files during processing, regardless of whether **Active** or **Refresh during document processing** are enabled in the plan file.

The plan file selection on the **Plan Files** tab affects the Axiom query list as follows:

- If you have selected individual plan files, then only the eligible queries for the source templates of the selected plan files are shown.
- If you have selected **All** or **Use Filter**, then all eligible queries for all used templates are shown. If the file group has templates that have not been used to create any plan files, then those templates are not included in the list.

The listed queries are identified by template, worksheet, and query name. The following additional properties are also listed for each query:

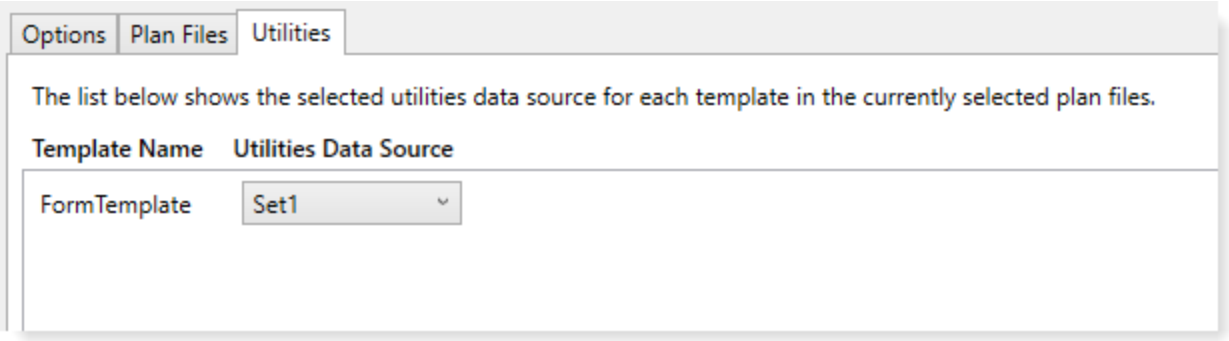
- **Refresh On Open**: Indicates whether the Axiom query is configured to refresh automatically when the file is opened. This is for information purposes only, to help you determine whether the query needs to be included in the processing. The Refresh on Open status is ignored by Process Plan Files—if the query is selected it will be run along with the other selected queries, and if it is not selected it will not be run.

- **Dynamic**: Indicates whether the query is dynamically enabled. True means that the query uses a formula for the **Active** setting.

**NOTE:** If a query is listed on this tab but it is grayed out and unavailable for selection, that means that although the query is active (either directly or dynamically), the query is not eligible to be run using Process Plan Files (because the setting **Refresh during document processing** is set to **Off**). This query is listed for your information only, so that you understand the query cannot be run as part of the process.

## Utilities tab

On the **Utilities** tab, select the ProcessPlanFileUtilities data source to use during processing. This data source determines which utility files are processed and the processing order. This tab only applies when using **Process with Utilities** mode.




*Example Utilities tab*

For each template listed, use the **Utilities Data Source** field to select the data source to use for plan files created from that template.

- If the template only has one data source, that data source is selected.
- If the template has multiple data sources, then the data source marked as the default data source is selected by default. If desired, you can use the drop-down list to select a different data source.

When plan files are processed, Axiom reads the specified data source in each plan file to determine the utilities to be processed for that plan file.

The plan file selection on the **Plan Files** tab affects the Utilities list as follows:

- If you have selected individual plan files, then only the templates used to create the selected plan files are shown.
- If you have selected **All** or **Use Filter**, then all used templates are shown. If the file group has templates that have not been used to create any plan files, then those templates are not included in the list.

# Controlling access to plan files

User access to plan files in a file group is controlled by Axiom Security. Step ownership in a plan file process can also impact the level of access that a user has to a plan file at a particular point in time. This topic summarizes how plan file access is controlled by security and plan file processes.

## Plan file access

For a user to be able to see and open plan files in a file group, they must have plan file permissions defined in Security, on the **File Groups** tab.

For each file group, a user can be assigned an access level (**No Access**, **Read Only** or **Read/Write**) and a filter that determines which files the access level applies to. Both the access level and the filter can be configured at the user level and/or inherited from a role. Users can also have multiple permission sets for a file group, with different access levels applying to different sets of plan files. Additional options within the permission set determine what a user can do within the plan files, such as whether the user can save data or insert a calc method.

When a user opens the **Open Plan Files** dialog for the file group, the list of plan files is limited to only show the files that the user can open with either read-only or read/write permission. When the user selects a plan file to open, it is opened according to the user's access level to that particular plan file. The No Access permission is effectively ignored in this context; plan files set to this level of permission do not display in Open Plan Files. No Access is only used in conjunction with plan file processes (see below) or when using "combine" role inheritance (with the intent of the combined permission resulting in a higher level of access).

Generally speaking, if a user does not have access to any plan files in a file group, then the user will not see that file group in ribbon tabs, task panes, and other areas of Axiom. Even if the file group displays to the user (such as by using **Show Restricted Item** in a task pane), the user will not be able to open any plan files in that file group.

## Ownership in a plan file process

The second level of control is ownership of the plan file in a plan file process. A plan file process is an optional feature of process management that allows you to define sequential planning steps for plan files. For each step in the process, an owner is assigned to each plan file, to carry out the task of either editing or reviewing the file. If a user is the owner of a plan file for a current process task, then Axiom will "elevate" the user's permissions as necessary to allow the user to complete the task.

Using this approach, it is possible to configure a setup where a user has no access to a plan file unless they are the current owner of the file in a plan file process. If the user has a permission set with **No Access** and **Interacts with Process Management**, then under normal circumstances that user cannot see or open the plan file. However, when the user is the current owner, the user's permissions will be temporarily elevated as appropriate so that the user can complete the task (for example, the permission

would be elevated to **Read/Write** and **Allow Save Data** for an edit task). While the task is active, the user can open and edit the plan file, and save data from it. When the user completes the task, then the user is no longer the owner and the user's permissions would revert back to no access.

Ownership in a plan file process can only elevate existing user permissions, it cannot reduce or remove user permissions. If a user has been granted read/write permission to a plan file in security, then that user will always have that permission, regardless of whether they are an assigned owner in a plan file process. For more information, see the *Plan File Process Guide*.

# Restoring plan files from restore points

You can restore one or more plan files as of a particular point in time (a "restore point"). This feature is intended to ease the process of "rolling back" plan files if a process is run by mistake or if a change has unintended impacts.

For example, imagine that someone runs Create Plan Files and accidentally overwrites 10 plan files that should not have been overwritten. You can use the restore utility to restore all 10 plan files to the state they were in before they were overwritten.

This feature is limited to administrators and to users with one of the following security permissions: **Administer File Groups** (for all file groups) or **Modify File Group** (for specific file groups). Users with the Administer File Groups permission can restore all plan files for all file groups, even if the user does not otherwise have access to those files. Users with the Modify File Group permission can only restore plan files for which the user has read/write access.

**NOTES:**

- This process is intended to restore plan files that were changed by a bulk process such as Create Plan Files or Process Plan Files. If you want to restore a prior version of an individual plan file, see the *System Administration Guide*.
- This feature does not apply to file groups that use virtual plan files. In this configuration, physical plan files do not exist and therefore cannot be restored.

## How restore points are created

Restore points can be created in various ways:

- You can manually create a restore point for plan files at any time. For more information, see Managing restore points.
- You can optionally create a restore point before processing plan files. For more information, see Process Plan Files dialog.

- Certain features in Axiom automatically create a restore point before performing their activity. These features include:
    - Create Plan Files (if existing plan files are overwritten)
    - Update Persistent Plan Files
    - Apply Calc Method Changes
    - Archive Plan Files

## Performing a restore

When restoring plan files, you first select a restore point and then you select the plan files to restore.

**NOTE:** If a plan file is deleted, it cannot be restored using restore points. Restore points can only restore earlier versions of existing plan files.

**To restore plan files:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Select the file group for which you want to restore plan files, then click the **Restore Plan Files** button in the toolbar. (This command is also available on the right-click menu.)

3. On the first screen of the **Restore Plan Files** dialog, select the restore point to use, and then click **Next**. Restore points are identified by date/time and description. System-generated restore points list the name of the associated process in the description.

4. On the second screen of the dialog, select the plan files to restore, and then click **Finish**.

   This screen only lists the plan files that have been changed since the date/time of the selected restore point. You can select individual plan files or you can select the check box in the header to restore all of the plan files. You can use Axiom grid features to sort and filter the list.

   If no plan files have been changed since the selected restore point, a message informs you of this.

The restore process creates a new plan file version that is a copy of the version from the selected restore point, and that new version becomes the active version. For example, imagine that the current plan file version is version 10, and you want to restore version 9. The system creates version 11 of the plan file, which is a copy of version 9. Both version 9 and 10 still exist as independent versions in the document history for the plan file.

## Managing restore points

You can create custom restore points and you can delete existing restore points that are no longer needed. For example, you may be preparing to open the file group for end user input, and you want to manually create a restore point before users start editing plan files.

**To manage restore points:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Select the file group for which you want to manage restore points, then click the **Manage Restore Points** button in the toolbar. (This command is also available on the right-click menu.).

3. Manage restore points as desired:
   - To create a new restore point, click **New**. Enter a description for the new restore point, and then click **OK**. The date/time for the new restore point will be the current date/time.
   - To delete a restore point, select the restore point and then click **Delete**. This action simply deletes the restore point so that it is no longer available for selection; it does not delete the associated plan file versions from the database audit history.

     **NOTE:** Restore points are automatically deleted when the plan file versions associated with the restore points are no longer available in the database audit history. See the following section for more information.

### Availability of plan files to restore

The availability of restore points and the associated plan files depend on your system purge settings, as defined in the Scheduler system job **System.SystemDataPurge**.

The setting **Audit History: Number of days to keep system history** determines how long old plan file versions are stored in the database. For example, if this option is set to 15 days (the default setting), then you can only restore plan files from the last 15 days; anything older than that has been purged from the database. Therefore, Axiom also uses this setting to automatically delete old restore points from the same time frame, since those restore points no longer have any plan files available to restore.

# Copying plan files to other file groups

You can copy plan files from one file group to another. This feature is typically used to support specialized solutions for requirements such as:

- When plan files must be moved from one file group to another, to continue planning for them in a different file group—for example, when a capital request spans multiple years and must be copied to the next year of planning.
- When you want to use a plan file from one file group as a "starter template" for a new, similar request in a different file group.

The available features for copying plan files between file groups depend on whether the file groups are standard file groups or on-demand file groups. It is not possible to copy plan files between a standard file group and an on-demand file group—both source and target file groups must be the same type.

## Copying between standard file groups

For standard file groups, plan files can be copied manually using the **Copy Plan Files** feature. This feature is only available to administrators and to users with the **Create Plan Files** security permission. Using this feature, plan files can be copied in bulk to a target file group.

You can copy plan files to any file group that shares the same plan code table. It is up to the solution designer to ensure that the file groups are set up appropriately so that the copied plan files work as expected in the new file group.

**IMPORTANT:** The target file group must contain templates with the same names as the templates used to create the original plan files, or else the copy action will not be allowed. When the plan files are copied, the copies will refer to the templates in the target file group as their source templates, and those source templates will be used to provide lists of Axiom queries for Process Plan Files, and lists of print views for Print Plan Files.

**To copy plan files to another file group:**

1. In the Explorer task pane, expand the file group that you want to copy plan files from, and then double-click **Copy Plan Files**.

   The **Copy Plan Files** dialog opens. By default, the **Source File Group** is set to the file group in which you launched Copy Plan Files.

2. For **Destination File Group**, select the file group that you want to copy the plan files to.

   You can select any file group that uses the same plan code table as the source file group. If no file groups are listed, then no eligible file groups are available.

3. Select the plan files that you want to copy to the destination file group.

   You must have at least read access to the files in the source file group, and read/write access to the files in the destination file group.

   You can select individual plan files, or select the check box in the header to select all plan files. You can also use Axiom grid features to filter the list. If you select the check box in the header after filtering the list, only the currently-displayed plan files will be selected.

4. Click **OK**.

   A confirmation dialog informs you of the number of plan files that will be created or overwritten in the destination file group. Click **OK** to continue.

The selected plan files are copied to the destination file group. If you want those copied plan files to be refreshed with data using the settings of the new file group and their data saved to the database, then you must use Process Plan Files.

**NOTE:** If needed, you can configure a plan file process so that plan files can be automatically copied to a target file group as part of completing a process step.

### Copying between on-demand file groups

For on-demand file groups, the **Copy On Demand Plan Files** feature can be used to copy plan files between file groups. This feature is available as follows:

- The Copy On Demand Plan Files command in the Command Library can be used in a custom task pane or ribbon tab to provide copy functionality to the intended users. For more information on this command, see the Axiom Help files.
- The Copy On Demand Plan Files task in Scheduler can be used to copy plan files as part of executing a Scheduler job. You can schedule copy operations as needed, and you can trigger copy operations using RunEvent (such as from an Axiom form). For more information on this task, see the *Scheduler Guide*.

When setting up the command or Scheduler job, you specify a source file group and a target file group. The copy feature does not enforce any relationship between the two file groups other than they must both be on-demand file groups. It is up to the solution designer to ensure that the copied plan files will operate as expected in the target file group.

The following is an example of how this functionality works:

1. The user executes the Copy On Demand Plan Files command from a custom task pane or ribbon tab.
2. A dialog displays a list of plan files in the source file group. The user can select one or more plan files to copy to the target file group.
3. When the user clicks **OK** to copy, one or more new records are created in the target file group, and the selected plan files are copied for use with those new records. Depending on the options enabled for the command, the new plan files may be saved after the copy in order to execute a save-to-database, or a data copy utility may be run in order to copy the necessary data.

When executing the command using Scheduler, in most cases you will specify a filter to determine the plan files to copy. This filter can be hard-coded in the task, or you can use job variables to dynamically change the plan files to copy.

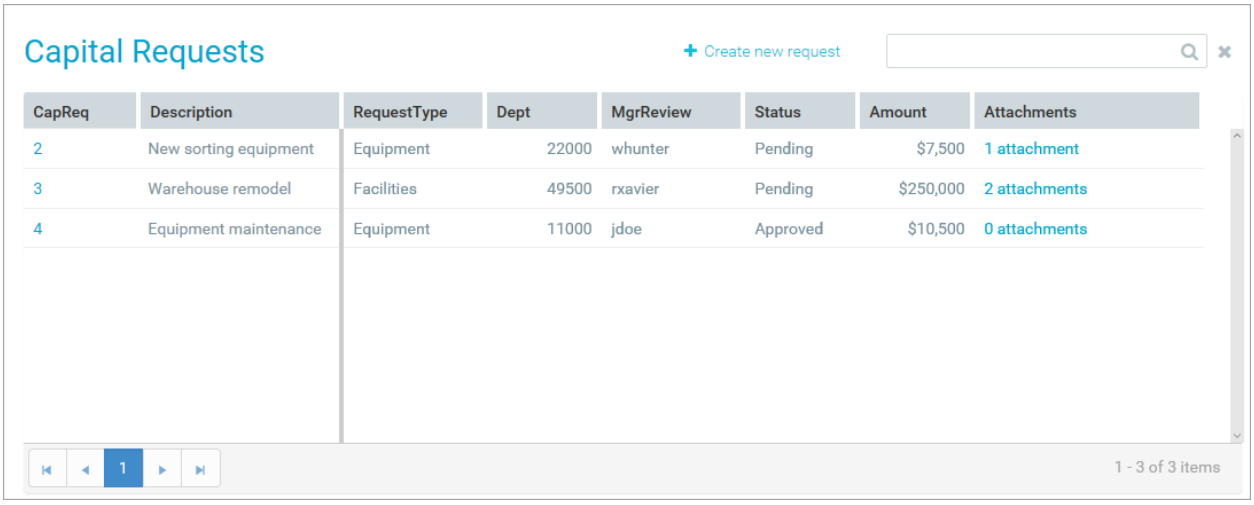
# Using the Plan File Directory page

The Plan File Directory web page is available in the Web Client and serves a similar purpose as the **Open Plan Files** dialog in the Desktop Client. You can direct users to this page, to provide them with an easy way of opening their available plan files in the web.

The Plan File Directory page shows a list of plan files that the user has permission to access, for a particular file group.

- You can customize this page to specify which columns are shown, the initial sort level of the list, and other formatting options.

- To open a plan file, users click the hyperlink in a designated column. In the following example screenshot, the CapReq column is the designated hyperlink column.
- Users can use the search box at the top to locate a particular plan file. You can configure which columns are included in the search.
- If plan file attachments are enabled for the file group, users can view and manage attachments from the directory using the optional **Attachments** column.
- You can also optionally provide users with a set of predefined options to filter the list, using refresh variables and the **Filters** panel.




*Example Plan File Directory page*

If the file group is an on-demand file group, users can also create new on-demand plan files by clicking the plus button in the top right-hand corner. This button uses the Add File Message text, and it only displays if the file group has a designated Add File Form (both as defined in the file group properties). Clicking the button launches the form. This is equivalent to the functionality in the Open Plan Files dialog to create a new on-demand plan file.

The Plan File Directory page is primarily intended for use in cases where plan files are form-enabled, and the end users' primary client is the Web Client. However, the page can also be used as an alternate means to open spreadsheet plan files.

For more information on customizing this page, see the *File Group Administration Guide*.

## Accessing the Plan File Directory page

There is no built-in way for users to navigate to the Plan File Directory page for a file group. If you want users to access this page, you should create a link to the page within one of the following:

- A custom task pane (either within the Desktop Client or a web navigation task pane)
- A custom ribbon tab
- An Axiom form

The Plan File Directory command can be used in these assets to automatically open the Plan File Directory page for a file group. When setting up the command, you specify the target file group and whether the directory should be opened in a new window. When the command is used, the appropriate directory page is automatically opened.

Although the command is the preferred method to open the directory, you can also manually generate a URL to the directory page and use it as needed. To create a URL to the Plan File Directory page for a file group, use the following syntax:

```
<baseURLtoAxiom>/FileGroups/FileGroupID/Directory
```

For example, if the file group ID is 50, the URL would look as follows:

```
https://ClientName.axiom.cloud/FileGroups/50/Directory
```

You can use the GetFileGroupID function to return the ID for a file group. To make the link dynamic, you can use a file group alias in the function. For example:

```
=GetFileGroupID("Current Budget")
```

Where Current Budget is the name of a file group alias that always points to the file group for the current budget cycle.

## Configuring display settings for the Plan File Directory web page

You can optionally configure various display settings for the Plan File Directory web page. You can configure the following on a per file group basis:

- The header text that displays on the top of the page
- Which columns are included in the directory, and certain attributes about those columns—including which column contains the links to the corresponding plan files
- Which column is used to sort the directory
- Whether users can filter the directory based on refresh variable selections
- The text for the "add new file" button (on-demand file groups only)

These settings are defined in the file group properties. Only administrators and users with one of the following security permissions can edit file group properties: **Administer File Groups** and **Modify File Group**.

**To access the Plan File Directory properties:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. Navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

   **TIP:** You can also do this from the file group node in the Explorer task pane.

Most of the properties that affect the Plan File Directory page are defined on the **Web Configuration** tab, in the **Plan File Directory** sub-tab. After making and saving any changes to the file group properties, you must reload the web page to see the effects of these changes.

### Customizing the directory header text

You can optionally customize the header text that displays at the top of the Plan File Directory web page. By default, the header text is the display name of the file group.

To customize the header, type the desired text into the **Directory Header Text** box on the **Plan File Directory** tab. If you later want to revert back to using the default text, you can clear the contents of this box.

You can use file group variables such as `{FileGroupYear}` in the header text. You must manually type any variable that you want to use, enclosed in curly brackets—there is no helper tool available to insert these variables for you.

### Configuring the display columns for the directory

You can optionally configure the columns that display in the directory, as well as certain attributes about those columns. You can specify which columns are searchable, which are frozen for scrolling, and the column width. You can also customize the header text for the column.

**IMPORTANT:** The display column properties determine which column contains links to the plan files. By default, no column contains these links. If you do not enable **Link to Plan File** for at least one column, then users cannot open plan files using the Plan File Directory page.

The current display columns are listed in the **Display Columns** box on the **Plan File Directory** tab. By default, the key column of the plan code table is included, as well as any designated description columns. You can include any column from the plan code table, from a data table that directly looks up to the plan code table, or from a reference table that the plan code table looks up to.

You can also include various **Plan File Columns**. The **Attachments** column in this section enables attachment management for plan files from the directory. The column displays the current number of attachments for the plan file. Users can click this number to access the File Attachments dialog for the plan file. This is the same dialog used by the File Attachments command in Axiom forms, allowing users to upload, view, edit, and delete attachments (according to user permissions). The Attachments column only displays in the directory if plan file attachments are enabled for the file group.

To configure which columns are included and in what order, click **Select Columns**. In the **Select Columns** dialog:

- To add a column, select the column in the left-hand pane of the dialog and then click **Add** to move it to the **Selected Columns** box.
- To remove a column, select the column in the **Selected Columns** box and then click **Remove**.

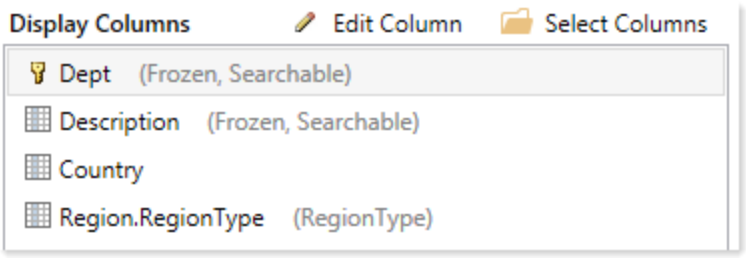
- To change the order of a column, select the column in the **Selected Columns** box and then click **Up** or **Down** to move it to the desired location.

Display attributes for each column are configured after the column has been added to the **Display Columns** box. To configure the display attributes for a column, select the column in the list and then click **Edit Column**. You can edit the following display properties:

| Item | Description |
|---|---|
| Header | The header text for the column. By default, this is the column name. You can customize this text if desired.<br><br>If the column is not on the plan code table, the fully qualified name is used by default. For example, if the plan code table is Dept, then if you add the `Dept.Region` column the default header value is just Region. But if you add the `CapData.Total` column, then the default header value is CapData.Total. |
| Width | The width of the column. By default, all display columns use the same width. If desired, you can enter a different width in pixels.<br><br>If you want to go back to using the default width, you can clear this field. |
| Column Alignment | The alignment of the column values. By default, the alignment is set to **Default** and determined as follows:<br>• Values in frozen columns are left-aligned.<br>• Values in non-frozen columns are left-aligned for strings and right-aligned for numbers.<br><br>If desired, you can override this behavior and specify the alignment as **Left**, **Right**, or **Center**. If you change the alignment and then you later want to return to the default behavior, specify Default.<br><br>**NOTE:** The alignment only affects the values in the column. Column header text is always left-aligned. |
| Searchable | Specifies whether the column is searchable on the web page, using the search box on the top of the page. Select this check box if you want the contents of this column to be included in the search.<br><br>If no columns are flagged as searchable, then the search uses the frozen columns.<br><br>**NOTE:** This option is only available for columns on reference tables. |
| Frozen | Specifies whether the column is "frozen" on the page for scrolling purposes. Select this check box if you want this column to remain fixed on the left-hand side of the screen when the user scrolls to the side. |

| Item | Description |
|---|---|
| Link to Plan File | Specifies whether the values in this column contain hyperlinks to the corresponding plan files. Users can use these hyperlinks to open their plan files.<br><br>This must be enabled on at least one column in order to allow users to open plan files from the Plan File Directory. Typically this option is enabled for the key column (such as Dept) or for another column that holds identifying values for plan files. For example, if the file group is an on-demand file group, you may be using an alternate key column that contains meaningful codes for each plan file, and you may be using that column instead of the identity column as the primary identifier for plan files.<br><br>**NOTE:** This option is not available for the Attachments column. The values in the Attachments column contain hyperlinks to the File Attachments dialog. |
| Custom Formatting | If the column values are numeric—meaning column data types of Integer (all types) or Numeric—then you can optionally define a custom display format for the values.<br><br>To define a display format, enter a valid Excel formatting string. These strings can be obtained as follows:<br>• Format a cell in a spreadsheet to use the desired display format.<br>• In the **Format Cells** dialog, on the **Number** tab, select the **Custom** category and copy the string in the **Type** box.<br><br>For example, this is the formatting string for a Currency format that shows the negative numbers in parentheses: `$#,##0.000_);($#,##0.000)`<br><br>Colors (such as red font for negative numbers) are not supported. Additionally, text replacement strings are only supported for zero values. Other advanced or unusual formats may not display as expected, so be sure to verify the column display.<br><br>If you do not define a custom display format, then the default formatting for the column's specified numeric type will be used. |

If you edit the attributes for a column, some of these changes display in parentheses after the column name in the Display Columns box. This is so that you can see certain attributes at a glance without having to open the Edit Columns dialog for each column. For example, in the following screenshot you can easily see which columns are searchable.




### Defining the sort column for the directory

You can optionally configure the column used to initially sort the directory. The user can change the sort by clicking on the column header of any column displayed in the directory.

The current sort column is displayed in the **Initial Sort Column** box on the **Plan File Directory** tab. By default, the directory is sorted by the key column of the plan code table, in ascending order. You can select any column in the plan code table, or in a reference table that the plan code table looks up to.

To configure the sort column, click **Select**. Then in the **Column Chooser** dialog, select the column that you want to use to sort the directory.

### Defining refresh variables for the directory

You can optionally set up refresh variables for the directory, so that users can filter the plan files shown in the directory based on these predefined variables.

To enable filtering for the directory, do the following:

- Create a file group utility file with a RefreshVariables data source, and define the variables to be used with the directory.
- Then, designate this file as the **Refresh Variable Workbook** for the file group.

When the Plan File Directory page is accessed, Axiom reads the variables from the designated file, and presents them in the Filters panel (just like when using refresh variables with Axiom forms). The user's selected values for the variables are applied as filters to the directory.

Refresh variables work as follows in this context:

- The only supported refresh variable type is ComboBox, using either a table column in the plan code table, or a ComboBox data source. All other variable types will be ignored.
- If you are using a column in the plan code table, Axiom takes the selected value for the refresh variable and applies it to the web page as a filter. For example, if the plan code table has a column such as RequestType, and the user selects Type1, then Axiom applies a filter of `RequestType='Type1'`. The web page is then filtered to only show the results for plan files that have a request type of Type1.

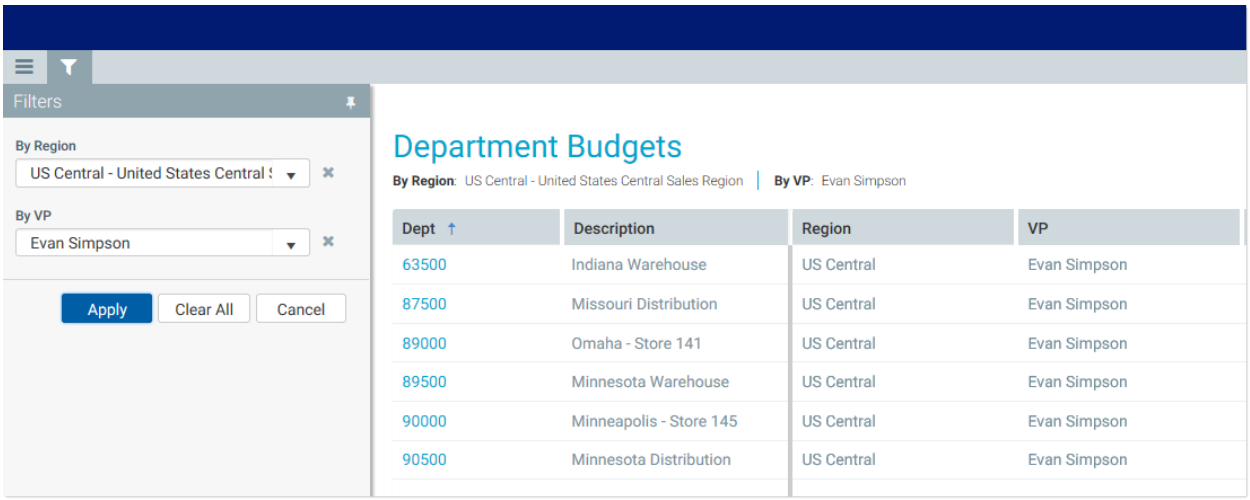
- If you are using a ComboBox data source, the `[Value]` column for the data source must contain valid filter criteria statements based on the plan code table. The selected filter is applied to the web page "as is". The `[Label]` column of the data source can contain "user friendly" text instead of the full filter statement.
- All other refresh variable settings can be used in this context, such as dependent variables or group names. Keep in mind that if you configure a variable as required, then once a user selects a value for that variable and applies it, they will not be able to clear the variable and return to the unfiltered state of the report without reloading the page.

The variables file can use Axiom queries, data lookups, and Axiom functions to define the variable properties or to populate the ComboBox data source. When the file is accessed by the web page, any "refresh on open" queries are executed and formulas are calculated before the variables are read from the file and presented in the Filters panel. When a user applies the variable values, the file is refreshed and calculated again before the final selected values are applied as filters to the web page. The file should not contain any queries or other features that are not necessary to the configuration of the refresh variables.

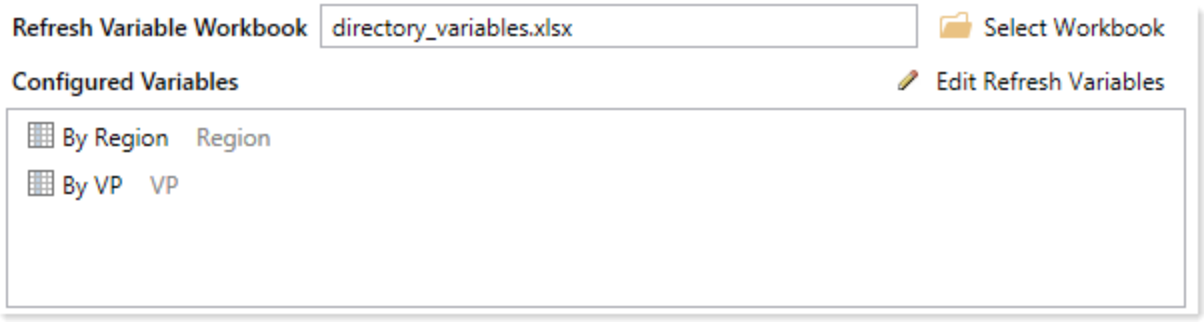
For example, the following variables could be defined in the utility file:

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | | | | | | | | | | |
| 4 | | | [RefreshVariables] | [Name] | [DisplayName] | [VariableType] | [IsEnabled] | [SelectedValue] | [IsRequired] | [ColumnName] |
| 5 | | | [Variable] | Region | By Region | ComboBox | TRUE | | False | Dept.Region |
| 6 | | | [Variable] | VP | By VP | ComboBox | TRUE | | False | Dept.VP |

When a user views the Plan File Directory page, they can use the Filters panel to filter the list by the variables:



To designate a file as the **Refresh Variable Workbook**, click **Select Workbook** on the **Plan File Directory** tab. You can select any utility file for the file group. Once a file has been selected, the variables defined in that file display in the **Configured Variables** box for your reference. If you want to edit the refresh variables, clicking **Edit Refresh Variables** opens the file.




## Defining text for the "add new file" button

If the file group is an on-demand file group, the Plan File Directory page contains a button for users to create new plan files. This button is only present if an **Add File Form** has been specified on the **Options** tab of the file group properties. When the user clicks the button, the designated form opens so that the user can create a new plan file.

You can customize the text that displays next to this button, by using the **Add File Message** field on the Options tab of the file group properties. This text is used in the Open Plan Files dialog for the Desktop Client, and in the directory pages for the Web Client.

10

# File Group Scenarios

Using file group scenarios, you can create additional scenarios or versions of your file groups, to perform activities such as:

- Development of alternative business scenarios
- Rapid "what if" analysis
- Sensitivity analysis
- Plan versioning

For example, you might have your baseline plan and then also have two scenarios for that file group—one with best case assumptions and one with worst case. You can report on these scenarios and compare the data.

The process of creating a scenario is similar to cloning a file group. A copy is created of the original file group, complete with templates, drivers, and plan files. Associated tables are also created to hold the scenario data. However, when you clone a file group you create a fully independent file group, whereas a file group scenario remains associated with its source file group.

The intent is that you will make modifications in the scenario—such as to change one or more key assumptions—and then compare the data in the scenario to the source file group to see the potential impact of this scenario on your plan.

## About scenarios

Using file group scenarios, you can create different scenarios or versions of a file group. You can modify drivers and assumptions in a scenario, save the data to the database, and then compare the data to the original file group in a report.

### Scenario name and table suffix

When you create a scenario, you define a scenario name and a table suffix.

- **Scenario Name**: The scenario name identifies the scenario. You can use names such as V1 or V2, or use more descriptive names such as Best Case or Worst Case.

- **Table Suffix**: The table suffix identifies the tables that hold the scenario data. Generally speaking, tables that hold assumptions, drivers, and planning data are copied for use in the scenario, so that the scenario data can be maintained separately from the original data. The suffix is limited to a maximum of 5 characters, so that table names are not excessively long.

For example, if you were to create a scenario for file group "2022 Budget," you could give it a scenario name of "V1" and a table suffix of "_V1". By default, the scenario would result in the following names and tables, as compared to the original file group:

| Item | Original File Group | File Group Scenario | Notes |
|---|---|---|---|
| File group name | Budget 2022 | Budget 2022 (V1) | Scenario names display in parentheses after the file group name. |
| Display name | Current Budget | Current Budget (V1) | The scenario name is also appended to the file group display name, if the file group has a defined display name. |
| Tab prefix | [BGT] | [BGT] | The tab prefix is unchanged, though you can edit it in the scenario properties after creation. |
| Writeable table variable for a reference table | BGT23_ Escalators | BGT23_ Escalators_ V1 | By default, the scenario clones all reference tables that the file group is configured to save data to. Table structure and data are copied. If you do not want a particular table to be copied, or if you do not want to copy data, then you can configure the table variable to override the default behavior. |
| Writeable table variable for a data table | BUD2022 | BUD2022_ V1 | By default, the scenario clones all data tables that the file group is configured to save data to. Only the table structure is copied. If you do not want a particular table to be copied, or if you want to copy data as well, then you can configure the table variable to override the default behavior. |
| Read-only table variable for a data table | GL2021 | GL2021 | If the file group is not configured to save data to a table (the table is for data queries only), then the scenario does not clone that data table. It continues to query data from the original table. |

## Scenarios and file group variables

In order to get the most out of the scenario feature, your file groups should use variables to define key assumptions and to identify the associated tables for the file group. The most important aspect of this process is to define and use table variables in the file group. When you create a scenario for the file

group, Axiom uses the table variable properties to determine which tables should be copied for use in the scenario.

For example, imagine that your file group has a table variable of `{PlanData}` that resolves to table name BGT2022. Then in the file group drivers and templates, you use the function GetFileGroupVariable to return this table name, to specify which table the plan files save budget data to.

When you create a scenario for this file group, Axiom copies the original file group and automatically adjusts the table variable in the scenario so that it now resolves to a table name of BGT2022_V1 (see the previous section for more information on how scenario names and table names are determined). Axiom also automatically creates this new table, by cloning the original BGT2022 table. Since the table name is referenced in drivers and templates by using GetFileGroupVariable, the plan files in the scenario automatically point to this new table, and the scenario is ready to be processed and save data as soon as it is created, without impacting your original file group.

If you do not use table variables in your file groups, then Axiom does not know which tables are associated with the file group, and therefore cannot create new tables for the scenario. Although you can still create scenarios in this case, you would be responsible for manually creating the necessary tables, and then manually adjusting the driver files and templates in the scenario to point to the appropriate tables, so that the data from the scenario does not overwrite the data in your original file group.

You can also use file group variables to define key assumptions for the file group. If you do this, and then reference those variables in the driver files, you can potentially perform all of your assumption changes when creating the scenario by changing the variable values. You could then process plan files for the scenario and report on data without ever having to open any files in that scenario. However, this is optional—if you aren't using file group variables to define key assumptions, then you simply need to modify the driver files in your scenario for the desired changes, and then process plan files for the scenario.

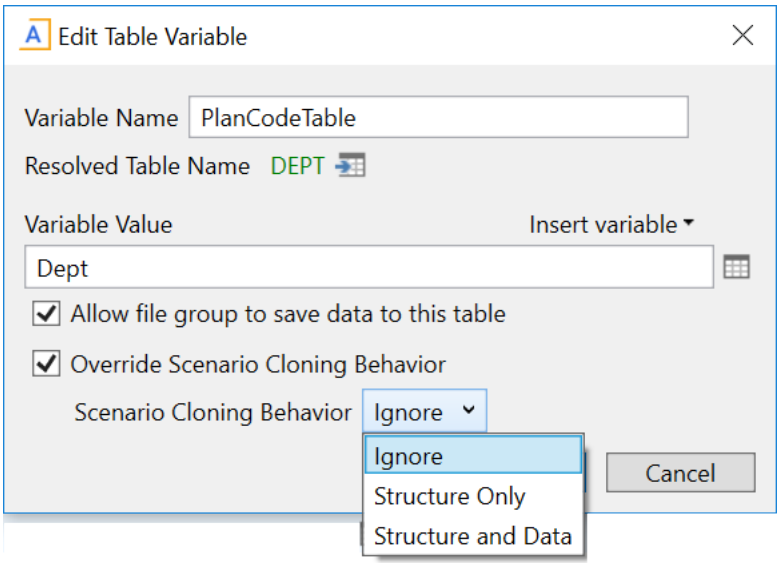
### Table cloning behavior for scenarios

When you create a scenario, new tables are created to hold the data for the scenario. This allows you to change assumptions and save updated data to scenario tables, without affecting the values in the original tables. You can then run reports that compare the data in the original tables to the data in the scenario tables.

Scenario tables are created based on the table variables in the file group. For each table variable where **Allow file group to save data to this table** is enabled, new tables are created as follows:

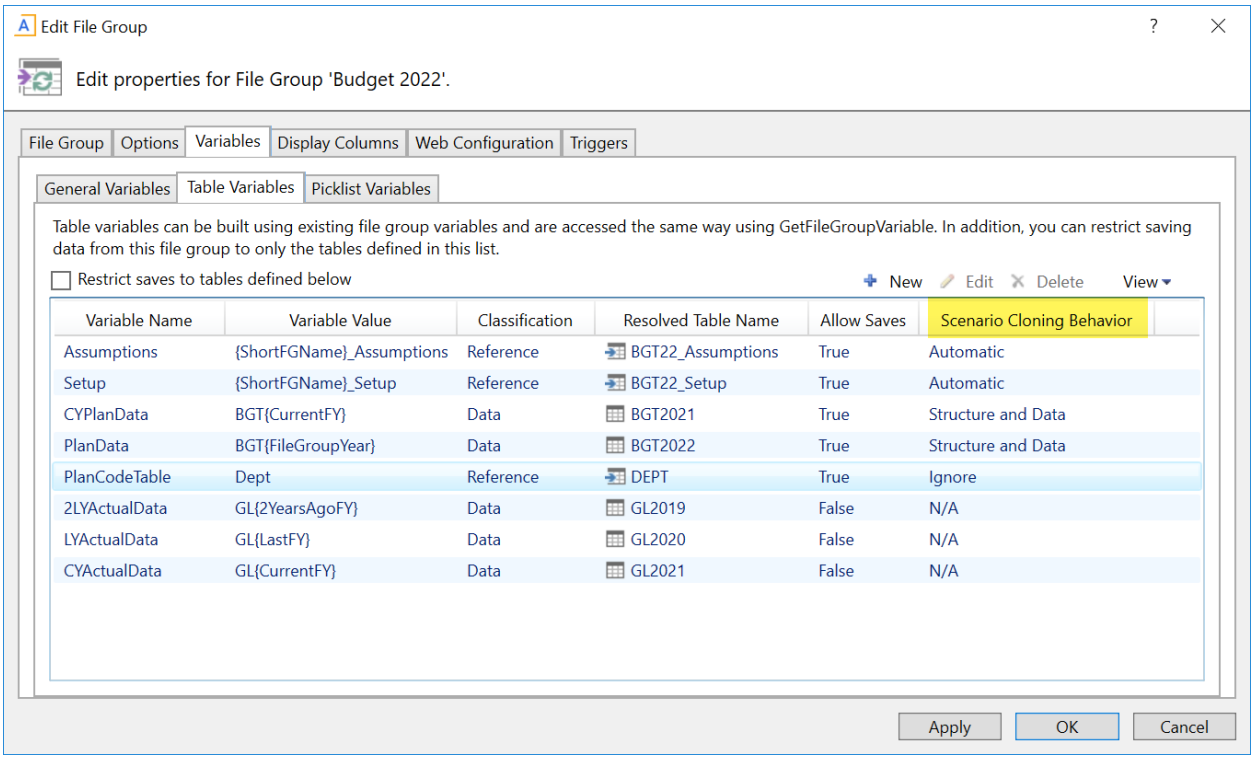
- A table suffix is added to each table variable value. For example, if a table variable pointed to BUD2022 in the original file group, it will point to BUD2022_V1 in the scenario.
- The original tables are cloned to create the new tables for the scenario. By default, reference tables copy table structure and data, and data tables copy table structure only.

- As needed, you can override the scenario cloning behavior for a particular table variable. For example, if the table should not be copied at all, then you can set the cloning behavior to **Ignore**. Or if you want to copy the data for a data table, you can set the cloning behavior to **Structure and Data**.




*Example table variable configured to override default behavior*

You can see the scenario cloning behavior at a glance when viewing the list of table variables. **Automatic** means the table uses the default cloning behavior, otherwise the configured override behavior displays.




*Example list of table variables showing the scenario cloning behavior*

Some tables are not cloned for scenarios and therefore display **N/A** as the scenario cloning behavior. For more information, see Setting up table variables for scenario creation.

# Setting up table variables for scenario creation

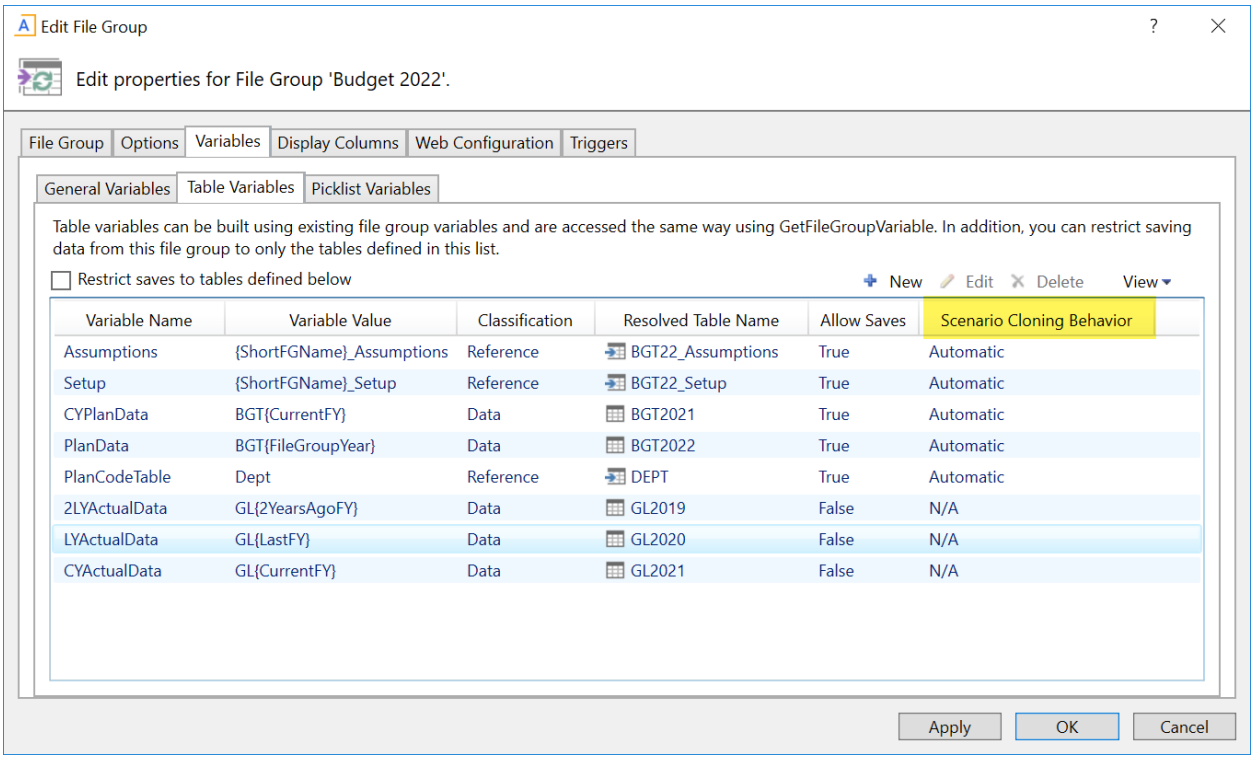
When you create a scenario, the table variables for the file group determine which tables are created for the scenario. Before attempting scenario creation, you should review these variables and configure them as needed.

## Default scenario cloning behavior

Scenario tables are created based on the table variables in the file group. For each table variable where **Allow file group to save data to this table** is enabled, new tables are created as follows:

- A table suffix is added to each table variable value. For example, if a table variable pointed to BUD2022 in the original file group, it will point to BUD2022_V1 in the scenario. The table suffix is defined when you create the scenario.
- The original tables are cloned to create the new tables for the scenario. By default, reference tables copy table structure and data, and data tables copy table structure only.

If a table uses the default scenario cloning behavior, it displays as **Automatic** in the list of table variables. In this example, none of the table variables have been configured to use a different behavior, so they all indicate either Automatic or **N/A**.




*Example table variables using default behavior*

Some tables are not cloned for scenarios and therefore display **N/A** as the scenario cloning behavior. This includes the following tables:

- If **Allow file group to save data to this table** is not enabled for a table variable, then its target table is not cloned. The idea is that these tables are queried only, and both the original file group and its scenarios need to query the same data.
- Document reference tables are not cloned, because document reference tables are created and edited via their source document. If you are using document reference tables for your driver tables, then these tables will be created by processing the driver files in the scenario once the scenario is created. Document reference tables are also known as "Save Type 3" tables.
- Tables with identity key columns are not cloned for the scenario.
- Picklist tables are not cloned, whether they are defined as a picklist variable or a table variable. Note that if the picklist table is defined as a table variable, it will display as if it uses Automatic or configured scenario cloning behavior, but it will be ignored when the scenario is created.

## Overriding the default scenario cloning behavior

In some cases, the default scenario cloning behavior is not appropriate for certain tables. For example:

- If the plan code table of the file group is the target of a table variable, then that table should not be cloned for the scenario. All scenarios for a file group must use the same plan code table. (Note that if the plan code table of the file group is an identity table, then there is no need to override the scenario cloning behavior because identity tables are automatically excluded from cloning when creating a scenario.)
- If the file group has table variables for other dimension reference tables besides the plan code table, these tables probably should not be cloned for the scenario. For example, if a table variable references an Entity table or an Account table, there is likely no need to create a copy of this table for the scenario.
- If the plan files in the file group are rebuildable—either virtual plan files, or persistent plan files designed as rebuildable—then tables that hold planning data should copy data as well as structure. Otherwise, any planning inputs made in the original file group and saved to the planning tables will not be available to the scenario.

If you need a particular table to use different scenario cloning behavior, then you can configure the table variable to override the default behavior. This change is made in the properties of the file group, before creating scenarios. For each table variable where **Allow file group to save data to this table** is enabled, you can override the behavior and choose one of the following:

- **Ignore**: The table is not cloned when creating a scenario. The scenario will continue to use the same table as the original file group.
- **Structure**: The table is cloned when creating a scenario, but only the structure is copied, not the data. Essentially, the table is empty until data is saved to it from the scenario.
- **Structure and Data**: The table is fully cloned when creating a scenario—both structure and data. This means that the scenario starts with the same data as the original file group.

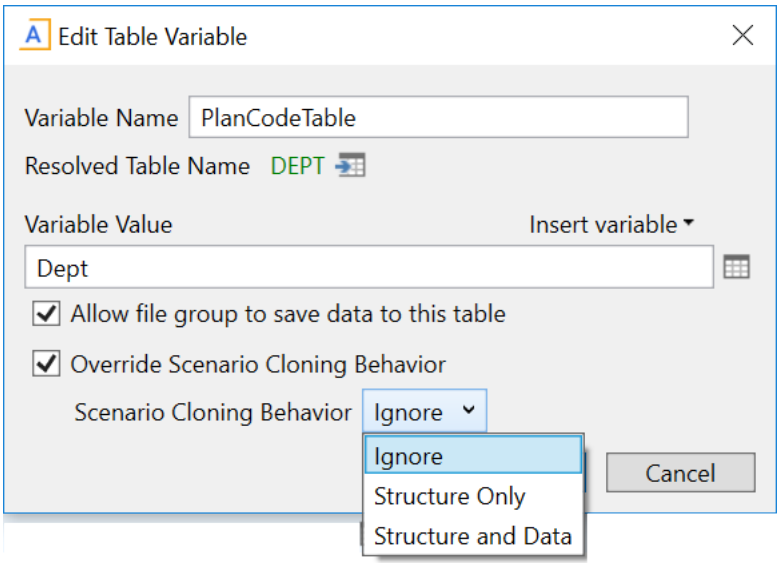
**To override the scenario cloning behavior for a table variable:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.
2. In the **Axiom Explorer** dialog, navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

   > **TIP:** You can also access the file group properties by right-clicking a file group in the Explorer task pane.

3. In the **Edit File Group** dialog, select the **Variables** tab, then select the **Table Variables** sub-tab.
4. Find the table variable that you want to edit, then double-click it.
5. In the **Edit Table Variable** dialog, do the following:

- Select **Override Scenario Cloning Behavior**. This exposes a drop-down list of cloning behavior options.
- For **Scenario Cloning Behavior**, select the desired behavior to use for this table—**Ignore**, **Structure**, or **Structure and Data**.




**NOTE:** If you ever want to restore the table variable to the default scenario cloning behavior, simply edit the variable to clear the **Override Scenario Cloning Behavior** check box.

6. Click **OK** to save your changes to the variable properties. You should see the updated cloning behavior for the variable shown in the grid.
7. Click **Apply** or **OK** in the Edit File Group dialog to save your changes.

# Creating a scenario

You can create scenarios for a file group to perform scenario planning. Each file group can have any number of scenarios.

When you create a file group scenario, you create a copy of the source file group as well as the necessary tables to hold the scenario data. This scenario "copy" remains associated with the source file group, but otherwise provides normal (although limited) file group functionality. You can perform actions such as modifying the scenario drivers, and then processing plan files to save the scenario data to the database. You can then compare the data between the source file group and the scenario in a report.

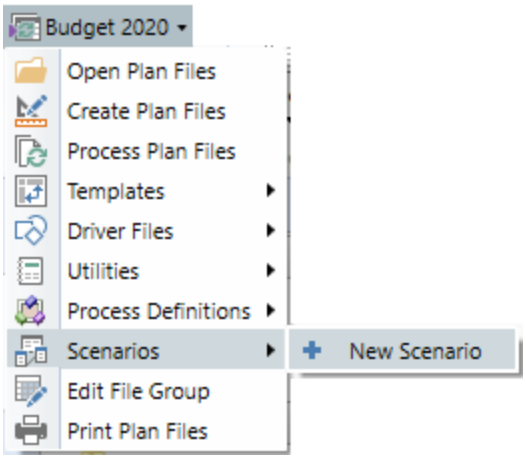
Before creating a new scenario, you should do the following:

- Decide on a scenario name and table suffix to identify the scenario in relation to its source file group. By default, the system uses scenario names such as "*FileGroupName* (V1)" to identify scenarios (where V1 is the scenario name). You can change this name when creating the scenario, but you should choose something that is short and descriptive. For more information, see About scenarios.
- Make sure all the writeable table variables for the file group are configured as needed to result in the necessary tables for the scenario. A "writeable table variable" means a table variable where **Allow file group to save data to this table** is enabled. For more information, see Setting up table variables for scenario creation.

**NOTE:** Only administrators and users with one of the following security permissions can create new scenarios: **Administer File Groups** (for all file groups) or **Modify File Group** (for specific file groups). If new data tables will be created as part of the process, the user must also have the **Administer Tables** permission. The user should also have permissions to all relevant documents such as driver files, to effectively use the created scenario.

**To create a scenario for a file group:**

1. On the **Axiom** tab, in the **File Groups** group, click the arrow next to the file group name to bring up the file group menu, then click **Scenarios > New Scenario**.



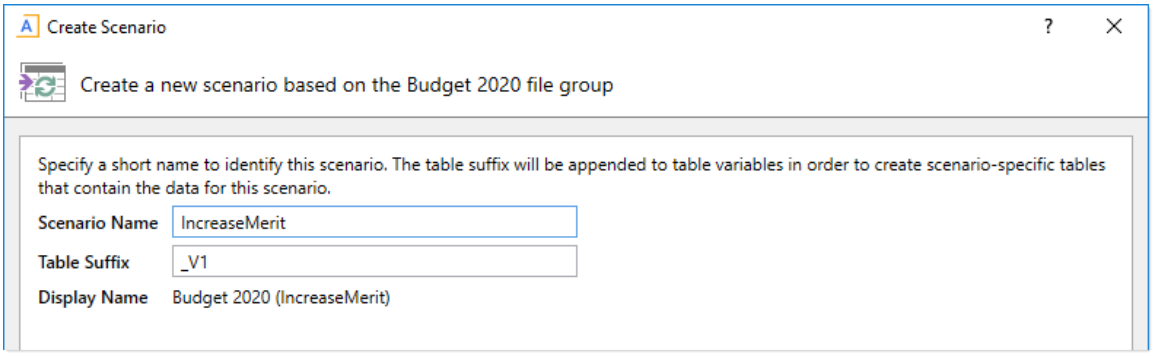

**TIP:** You can also create scenarios within Axiom Explorer and the Explorer task pane, by right-clicking the Scenarios node under a file group and selecting **Create Scenario**.

The **Create Scenario** dialog opens. This wizard will guide you through the process of creating a scenario.

2. On the first screen, complete the following items and then click **Next**.

| Item | Description |
|---|---|
| Scenario Name | The name of the scenario. By default, the scenario name uses the format **V#**, where V1 is the first scenario for the file group, V2 is the second, and so on. You can change the name if desired.<br><br>The scenario name is appended to the source file group name and the display name to identify the scenario. It is also used to create table subfolders to hold the scenario tables. The scenario name cannot be changed after the scenario is created. |
| Table Suffix | The suffix to apply to writeable table variables associated with this file group. By default, the suffix uses the same format as the default scenario name, with an underscore: **_V#**. You can change this suffix if desired.<br><br>The suffix will be used to create unique tables to hold the data for this scenario. The suffix is limited to 5 characters. |

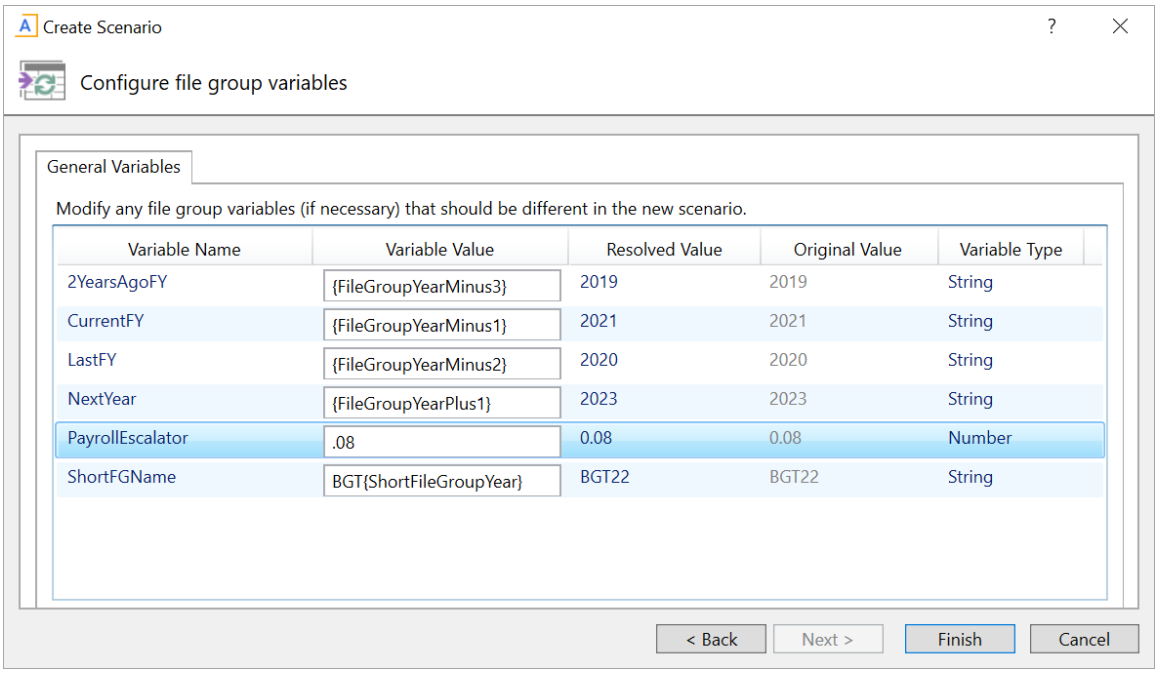
For more information on how scenario names and table suffixes are used, see About scenarios.




*Example scenario names*

3. On the **Configure file group variables** screen, review the general file group variables and make changes as necessary.

   If planning assumptions have been set up as file group variables, you may want to change those values for use in the new scenario. For example, you may have a variable for PayrollEscalator that defines a percentage increase for salaries, and you want to change this value in the scenario.




*Example file group variables*

4. Click **Finish** to create the scenario.
    - If any of the writeable table variables for the scenario resolve to existing data tables instead of new tables, a warning message displays and lists the tables. Review this list to make sure that the scenario should reference these original tables instead of creating new ones. Click **OK** to continue, or **Cancel** to stop the scenario creation. If any of these tables should instead be copied for use in the scenario, then you must edit the properties of the table variable to change its scenario cloning behavior. For more information, see Setting up table variables for scenario creation.
    - A confirmation message displays to inform you of the new tables that Axiom will automatically create as part of the scenario creation process. Review this list to ensure that new tables are being created as you expect. Click **OK** to continue, or **Cancel** to stop the scenario creation. If any of these tables should *not* be copied for use in the scenario, then you must edit the properties of the table variable to change its scenario cloning behavior. For more information, see Setting up table variables for scenario creation.

After the scenario has been created, a confirmation dialog informs you that the scenario was created successfully. From this dialog, you can choose to launch any of the following items, or click **OK** to close the confirmation dialog.

- **Configure scenario**: Opens the **Edit Scenario** dialog so that you can edit any of the file group settings for the new scenario. By default, the scenario inherited all settings that were not addressed in the Create Scenario dialog from the original file group. For more information, see File Group Properties.

- **Manage Driver documents and tables**: Opens the **Manage Drivers** dialog so that you can view driver properties for the new scenario and open driver files for editing. You may want to edit some driver values before performing Process Plan Files on the new file group.
- **Process Plan Files**: Opens the **Process Plan Files** dialog so that you can process plan files for the new scenario. If you made all necessary assumption changes via file group variables, then you may be ready to process plan files immediately after creating the new scenario.

These links are provided as a convenience, so that you can jump right into performing the next step for your scenario. You can access all of these features later using the normal menu paths.

### Scenario creation process

Axiom does the following to create the new scenario:

- The file group scenario is created by copying the existing file group. All files in the file group except process definitions and plan file attachments are copied. Security is also copied from the source file group to the scenario (see below for more information).

  **NOTE:** File group settings are copied in a similar manner as when cloning a file group. However, if the source file group is assigned to a category, the scenario will not be assigned to that category.

- For any table variables where the default or configured scenario cloning behavior results in new tables, those tables are created by cloning the original table. For example, BGT2022_v1 is created by cloning BGT2022.

  Newly created tables are placed in scenario-specific sub-folders. For example, if BGT2022 is located in `\Budget\Data`, then BGT2022_v1 is located in `\Budget\Data\V1` (where V1 is the scenario name).
- The driver files for the scenario are processed. This means that the driver files are opened, calculated, and saved. The save includes both a save-to-database and a file save. Axiom queries and data lookups are only run if they are set to refresh on open.

  If the driver files use Save Type 3 to create document reference tables, this process should result in a new set of driver tables for the scenario (ideally, using new table names as defined by the table variables).

## Scenario security considerations

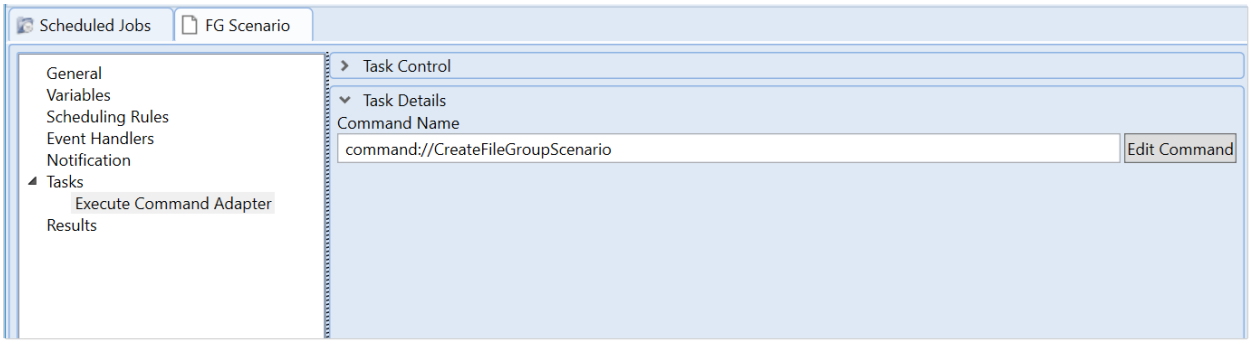
Once the scenario has been created, keep in mind the following security considerations:

- When the scenario is created, security settings are automatically copied from the source file group to the scenario. However, by default the scenario will not display on the Axiom ribbon tab or in the Explorer task pane, so the only users who can access it are administrators, and users with the **Administer File Groups** permission or the **Modify File Group** permission (for the source file group). If you want other users to be able to access the scenario, you must edit the Axiom ribbon tab (or another custom task pane or ribbon tab) to add the scenario.
- When new data and reference tables are copied for the scenario, they belong to the same table type as the original table and therefore inherit the table type security permissions. If the original table has table-specific security permissions defined, those permissions are copied to the new table. You can adjust this access if necessary.
- New document reference tables are treated as normal, meaning that all users have full read access via the Everyone role. You can adjust this access if necessary.

## Creating a scenario using Scheduler

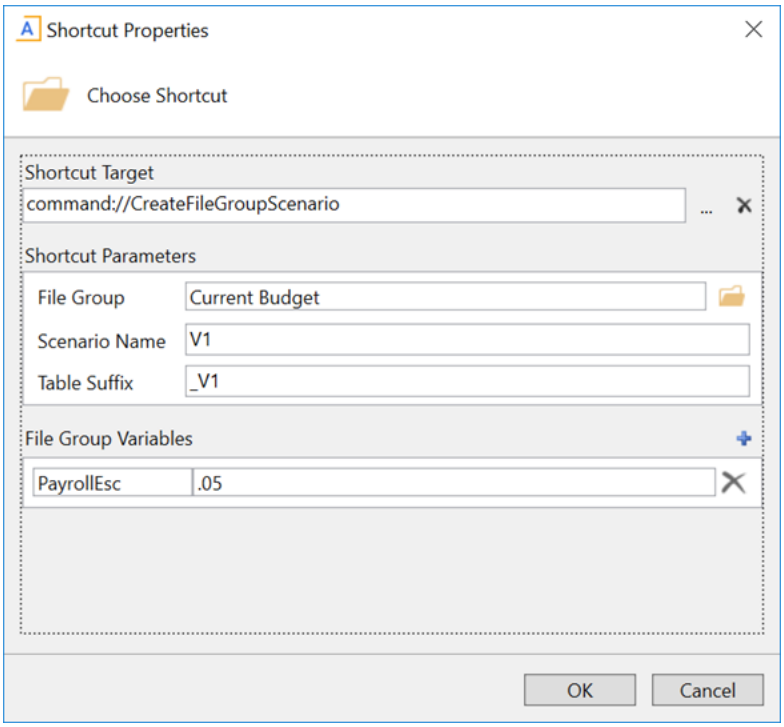
If desired, you can create a scenario using Scheduler instead of creating it interactively in the Desktop Client. By using Scheduler, you can schedule the creation for off-hours and leverage server-side processing. This may be preferable if the file group contains many files to be copied, or if many tables will be created for the scenario.

To create a scenario using Scheduler, create a job that uses the **Execute Command Adapter** task, and configure the task to use the **Create File Group Scenario** command.




*Example Scheduler job to create a scenario*

When you use the command, you can define a scenario name and a table suffix, and you can pass in values for file group variables. For example, if the file group uses a variable like PayrollEscalator, you can pass in a value for that escalator to be used in the scenario. Scheduler job variables can also be used to define the values for any of these fields.



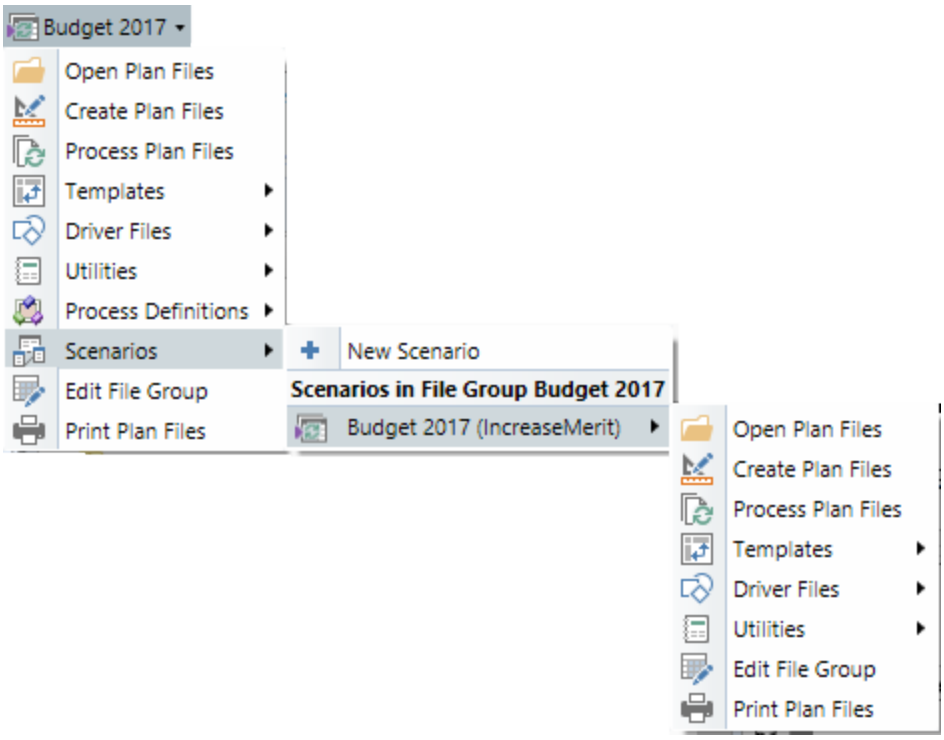

*Example shortcut properties for Create File Group Scenario command*

# Working with scenarios

Once you have created a scenario, you can work with it just like any other file group, with a few exceptions. The intended purpose of a scenario is to allow you to make modifications to key file group assumptions, and then save that data to the database so that you can compare it to the source file group. You can create as many scenarios as needed to perform the desired analysis.

## Accessing scenarios

Although scenarios are complete file groups with access to most file group functionality, they remain associated with their source file group. You can access scenarios using the **Scenarios** node of the source file group.




*Example ribbon menu*

The Scenarios node is limited to administrators and users with one of the following security permissions: **Administer File Groups** (for all file groups) or **Modify File Group** (for specific file groups). Other users will not see the **Scenario** menu option on the source file group. If other users need to access the scenario, then you can edit the Axiom ribbon tab (or any custom ribbon tab) to place it directly on the ribbon. This allows any user to access the scenario like any normal file group (assuming they have security permissions to the scenario file group).

## Common scenario activities

In most cases, the workflow for a scenario will be similar to the following example:

- You determine one or more assumptions that you want to model using a scenario. For example, what is the impact to the plan if salary increases are 4% instead of 2%?
- You create a scenario for that file group, and then modify the assumptions as necessary to model the desired change. You might do this by changing the value of a file group variable for the scenario, or by editing values within the driver files of the scenario (or both).
- You perform Process Plan Files on the scenario, to refresh the data for the changed assumptions and then save that data to the database. You can do this manually, or schedule it for processing using Scheduler.
- You create a report that compares the data in the source file group to the data in the scenario, to see the impact of your changed assumptions.

When you are done performing the analysis, you can delete the scenario or you can "promote" it to be a regular file group.

### Limitations of scenario file groups

The following features cannot be used on scenario file groups:

- Once a scenario has been created, you cannot change its name or its display name.
- Certain table variable properties cannot be edited in the scenario. If a table variable does not resolve to a scenario table, then you cannot edit the **Allow file group to save data to this table** property. Additionally, you cannot edit the scenario cloning behavior (this simply does not apply to the scenario, because you cannot create a scenario of a scenario).
- You cannot create a plan file process for a scenario file group.
- You cannot clone a scenario file group.
- You cannot create a scenario of a scenario file group (but you can create multiple scenarios of the source file group).

If you want to use any of these features on a scenario, then you must convert it to a regular file group (thereby breaking the association between the scenario and the source file group).

# Deleting a scenario

When you delete a scenario, all files related to that scenario will be deleted. Tables associated with the scenario are handled as follows:

- Document reference tables associated with a file in the scenario will be deleted along with that file.
- Data and reference tables that were created for the scenario will be automatically deleted, unless you opt not to delete them.

**NOTE:** This action cannot be undone. You should be sure that you no longer need the scenario before you delete it.

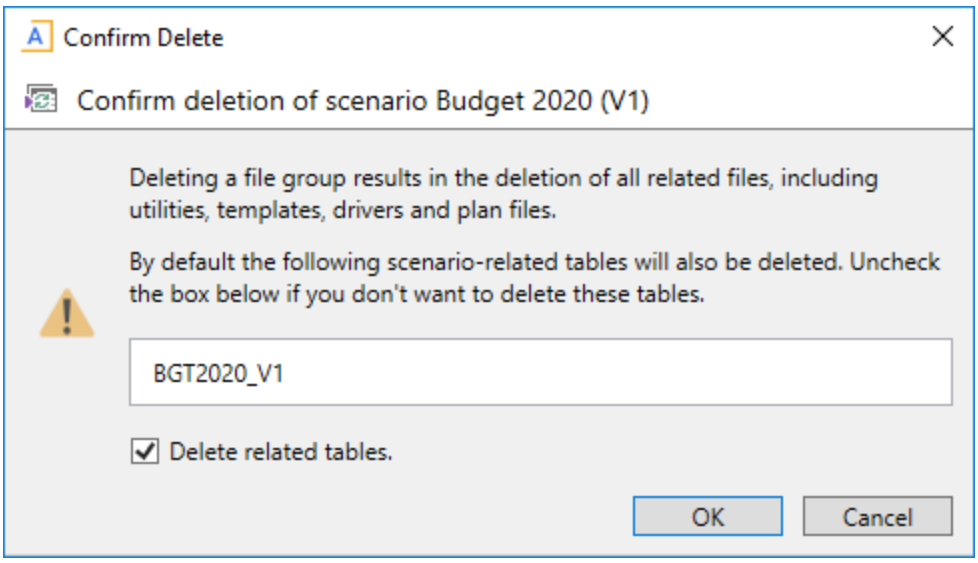
**To delete a scenario:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Navigate to the scenario that you want to delete, then right-click it and select **Delete**.

3. Review the confirmation prompt to confirm that you want to delete the scenario and the associated data tables. If you do *not* want to delete the tables, clear the check box for **Delete related tables**. Click **OK** to delete the scenario.




The scenario is deleted.

# Converting a scenario to a regular file group

You can "promote" a scenario to be a regular file group. When you do this, the scenario is no longer associated with its source file group, and all file group features become available for use.

**NOTE:** Promoting a scenario does *not* mean that the scenario replaces the original (source) file group. Both the source file group and the promoted scenario will exist independently, with no relationship between each other.

**To promote a scenario:**

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Navigate to the scenario that you want to promote, then right-click it and select **Promote Scenario**.

3. Axiom prompts you to confirm that you want to promote the scenario. Click **OK**.

The scenario is promoted to a regular file group. It no longer displays in the **Scenarios** node of the source file group.

11

# File Group Triggers

Using file group triggers, you can automatically "trigger" the execution of a Scheduler job after data is saved from plan files in the file group.

For example, you might have a report utility that you want to run automatically any time data changes in plan files. Or you might have one file group that is dependent on the data in another file group, so you want to process plan files in the second file group any time data changes in the first file group. To do this, you can set up a trigger on the file group, to cause Scheduler to perform the desired task whenever data is saved from plan files in the file group.

When you define a trigger for a file group, you specify:

- The name of the event to trigger when data is saved from plan files.
- One or more variables for the trigger, to be used in the triggered Scheduler job.

As part of the trigger setup, you must create the Scheduler job or jobs to be triggered. These Scheduler jobs must contain the event name for the trigger, and can optionally use variables that were set up for the trigger.

## How triggers work

Each file group can have one or more defined *triggers*, to trigger the automatic execution of Scheduler jobs after data has been saved from a plan file in the file group.

Each trigger in a file group has a defined event name. This event name is what links the trigger to the Scheduler jobs to be run. Additionally, triggers can use the following optional settings:

- One or more variables can be defined for the trigger. These variables can be used in the Scheduler job to impact the job execution in some way. The variables can be linked to a column in the plan code table, or they can have a defined value.
- Notification address settings can be defined for the file group. You can have a default email address for the file group and/or you can use a column in the plan code table that contains email addresses per plan code. The information passed to the triggered Scheduler job will be grouped by these addresses (meaning one job per unique address), so that you can use the {NotificationAddress} variable to send notifications.

Trigger events are triggered by any action that causes data to be saved from a plan file. For example, the triggering action could be a user who manually opens a plan file and then saves it, or a Process Plan Files utility that uses the save-to-database option. The event is only triggered if the save-to-database process results in a change to the database—if no data is changed in a particular plan file, then no trigger-related action occurs for that plan file.

If data is changed as a result of the save-to-database process, then the plan code for the associated plan file is added to a trigger queue. If the file group has multiple triggers, each trigger will be evaluated for that plan code.

If the plan file was saved independently, then the trigger queue is evaluated immediately. For each trigger in the queue, Axiom checks to see if any Scheduler jobs exist that contain an event handler with the same name as the trigger event. If any matching jobs are found, they are added to the Scheduler queue to be eligible for immediate processing (pending available Scheduler threads and any higher-priority jobs already in the queue). If no matching jobs are found, then no action is taken, and no error occurs. In both cases, the plan code / trigger combination is cleared from the trigger queue.

If the plan file was saved as part of a Process Plan Files utility, then the changed plan codes are still added to the trigger queue, but the trigger queue is not evaluated until the entire Process Plan Files utility is finished. Once the utility is complete, the trigger queue is evaluated, but in this case the plan code / trigger combinations are first grouped by notification addresses. This ensures that each unique notification address for the file group has its own Scheduler job, so that the {NotificationAddress} variable can be used to send notifications once the process is complete.

**Example**

Twenty plan codes are added to the trigger queue as a result of a Process Plan Files utility:

- If the notification address settings are not used, or if only the default address is used, then all twenty plan codes are evaluated as a single group. For each Scheduler job that matches the trigger event, one instance of that job is added to the Scheduler queue. All twenty codes will be considered for the evaluation of variables in that Scheduler job.
- If a table column is used for the notification address settings, and the twenty plan codes are associated with five unique email addresses, then the twenty codes are grouped into five different groups. For each Scheduler job that matches the trigger event, five instances of that job are added to the Scheduler queue. In each job, only the codes associated with that unique email address will be considered for the evaluation of variables in that Scheduler job.

The Scheduler job added to the Scheduler queue as a result of the trigger event can perform any Scheduler task. For example, it could be used to process a report, or to run an import, or to process plan files in a different, dependent file group. If the trigger event has defined variables, those variable values are available to the Scheduler job to impact the job processing.

**Example**

The trigger event has a variable named Region, which uses the DEPT.Region column. For each plan code in the trigger queue associated with this Scheduler job, the plan code's region is passed to the job, and is used wherever the variable {Region} is used in the job settings. If multiple plan codes result in multiple region values, then the list of regions is passed as a comma-delimited list.

If the Scheduler job was used to perform a Process Plan Files task on a different file group, it could be set up to use a filter such as `DEPT.Region IN ({Region})`. This would resolve to a filter value such as `DEPT.Region IN ('North','South')`, if the plan codes in the trigger queue which caused this Scheduler job to be triggered belonged to the North and South regions.

To prevent duplicate trigger processing, the following checks occur:

- If data is saved from a plan file, and the associated plan code / trigger combination is already in the trigger queue waiting to be evaluated, then it is not added to the queue again.
- When the trigger queue is evaluated and a matching Scheduler job is found, if an instance of that job is already in the Scheduler queue, then it is not added to the Scheduler queue again. The relevant passed values (variable values and notification address) are considered to help determine if a job instance is a duplicate.

# Defining triggers for a file group

You can define one or more triggers for a file group, to automatically execute Scheduler jobs after data is saved from plan files in the file group.

This section explains how to define triggers for a file group. In order for a trigger to perform any actions, the trigger must be enabled and one or more corresponding Scheduler jobs must have been set up for use with that trigger. For more information, see Setting up a Scheduler job for use with triggers.

Optionally, you can also set up file group-specific email notification addresses, to be used within the triggered Scheduler jobs. For more information, see Using email notifications with triggers.

Triggers are defined on the **Triggers** tab of the **Edit File Group** dialog. To open this dialog:

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Navigate to the file group that you want to edit, then right-click the file group and select **Edit**.

## Adding a trigger to a file group

1. At the bottom of the **Triggers** tab, click **Add Event**.

   A new trigger event is added to the Triggers tab.

**TIP:** You can also add an event by right-clicking in the tab.

2. Double-click inside the **Event Name** column so that an editable text box appears, and then type the desired name of the trigger event.

   This event name is what links the trigger to the Scheduler jobs to be automatically executed. Each Scheduler job that you want to be triggered must contain an event handler with the same name as the trigger event.

3. Optional. In the **Variables** column, create one or more variables for the trigger.

   - To add a variable, click **Add Variable**. (You can also add a variable by right-clicking the Variable Name header.)
   - To delete a variable, right-click the variable and then select **Remove Variable**.

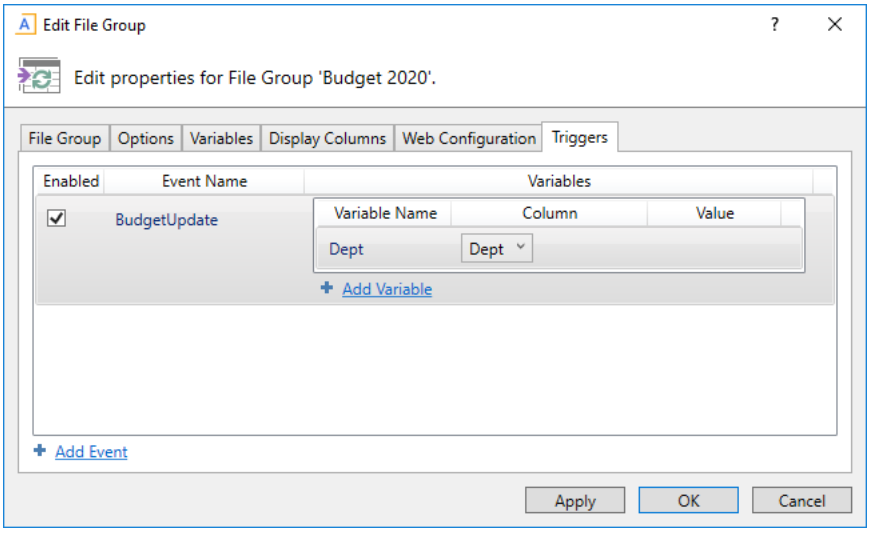
   Each variable must have a defined name, and then must be associated with a column in the plan code table or a fixed value. For more information, see Using variables with triggers.

4. By default, the **Enabled** check box is selected for the trigger event. If you are not yet ready to enable the trigger, clear the check box.

   If the trigger is enabled, no errors will occur if no matching Scheduler jobs are found—it simply means that no action will be taken as the result of the trigger.

5. Click **OK** to save.

The following is an example trigger event with a column-based variable:



## Editing a trigger in a file group

On the **Triggers** tab, you can edit any existing trigger setting. Keep in mind the following:

- If you disable a trigger, then that trigger will no longer be evaluated for the file group. Generally, this should be used to temporarily disable trigger processing. If you no longer need the trigger at all, you can delete it.
- If you enable a trigger, then that trigger will be active as soon as the file group settings are saved. You should be sure that the corresponding Scheduler jobs are finalized before enabling a trigger. No errors will occur if no matching Scheduler jobs are found—it simply means that no action will be taken as the result of the trigger.
- If you change the name of an event or a variable, you must also change the corresponding event handler name or variable name in any Scheduler jobs that are already set up for use with the trigger.

To edit an event name, variable name, or variable value, double-click the name. The name becomes editable, and you can type your changes directly in the grid. To delete a variable within a trigger, right-click the item and then click **Remove**.

## Deleting a trigger in a file group

On the **Triggers** tab, you can delete a trigger event if it is no longer needed. Select the trigger in the list, then right click and select **Remove Event**.

## Using variables with triggers

You can define one or more variables for use with triggers. These variables can be used in the triggered Scheduler jobs to impact the job execution in some way.

For each variable, you define the following:

| Item | Description |
|---|---|
| Variable Name | The name of the variable. To use the variable in the triggered Scheduler job, create a corresponding job variable with the same name and then use the variable in the desired task settings.<br><br>Each variable must be associated with one of the following:<br>• A column in the plan code table.<br>OR<br>• A fixed value. |

| Item | Description |
|---|---|
| Column | Select a column in the plan code table that contains the values for the variable. For example, you can select DEPT if you want to pass department numbers to the Scheduler job, or Region if you want to pass region names.<br><br>If multiple plan codes are evaluated for the trigger as a result of a Process Plan Files utility, then multiple column values will be returned as a comma-separated list. |
| Value | If you are not using a column, then enter a fixed value for the variable. Multiple values can be entered as a comma-separated list.<br><br>**NOTE:** Axiom passes the value to the Scheduler job in the same format as it appears here. If the value is to be used in a filter, you must place single quotation marks around it as appropriate. |

For example, you might create a variable named Region, which uses the column Region in the DEPT table. Imagine that as part of a Process Plan Files utility, 10 different plan codes save changed data to the database and therefore trigger a Scheduler job to be executed. If 7 of those plan codes belong to the North region and 3 of them belong to the South region, then the following variable value will be passed to the Scheduler job: `'North','South'`.

**NOTE:** The above example assumes that all of the plan codes are associated with the same email notification address (or that the optional notification address settings are not being used). If the plan codes were associated with different addresses, then a separate Scheduler job would be created for each unique address, and the variable values would be limited to the plan codes associated with each address.

For more information on how you might use a variable value in a Scheduler job, see Setting up a Scheduler job for use with triggers.

# Setting up a Scheduler job for use with triggers

File group triggers are used to automatically execute Scheduler jobs after changed data is saved from plan files in that file group. If you want to use a trigger, you must create one or more Scheduler jobs that are specifically designed for use with the trigger.

The Scheduler job can be set up to run any task, on any file or file group in the system. There are no limitations on what the Scheduler job can do once it has been triggered.

The Scheduler job must contain the following:

- An event handler with the same name as the trigger event name. The event name is what links the trigger to the Scheduler job(s) to execute.

- If the trigger has defined variables, and you want to use those variables to affect the execution of the Scheduler job in some way, then you must:
    - Create corresponding job variables in the Scheduler job, using the same names as the trigger variables.
    - Use the variables within the job settings to achieve the desired outcome.

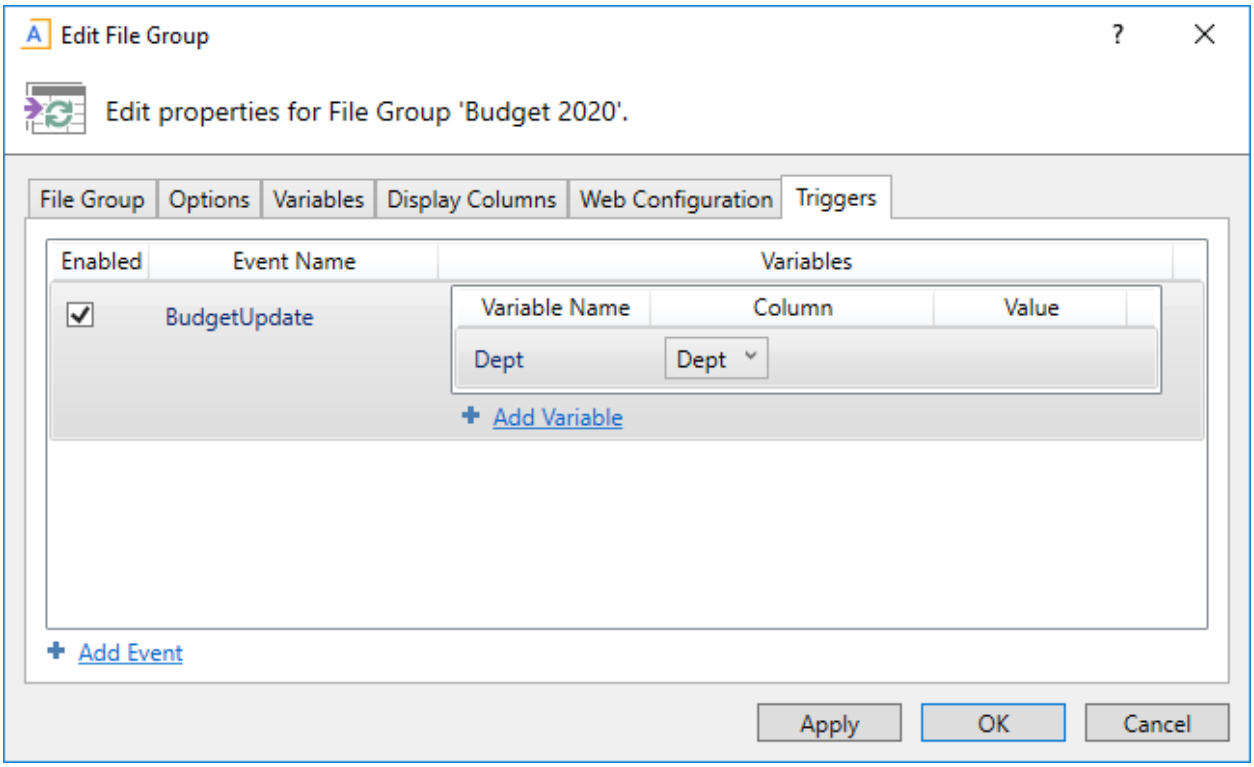
Optionally, you can also set up the job to use the notification addresses defined for the file group.

You can run any number of Scheduler jobs as the result of a file group trigger. When data is saved from plan files and the trigger events are evaluated, Axiom looks for any jobs that contain an event handler with the same name as the trigger event.

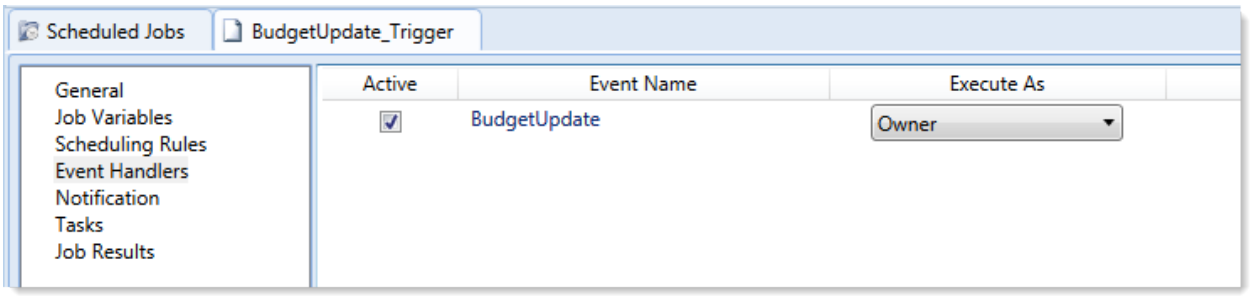
## Creating the event handler in the job

To associate the Scheduler job with the file group trigger, you must create an event handler within the job with the same name as the trigger event.

For example, if the name of the trigger event in the file group is BudgetUpdate, then the Scheduler job that you want to run as a result of that trigger must contain an event handler named BudgetUpdate.




*Example file group trigger event*




*Example Scheduler job event handler*

Both the file group trigger event and the Scheduler event handler must be enabled (active) in order for the Scheduler job to be executed as a result of the trigger.

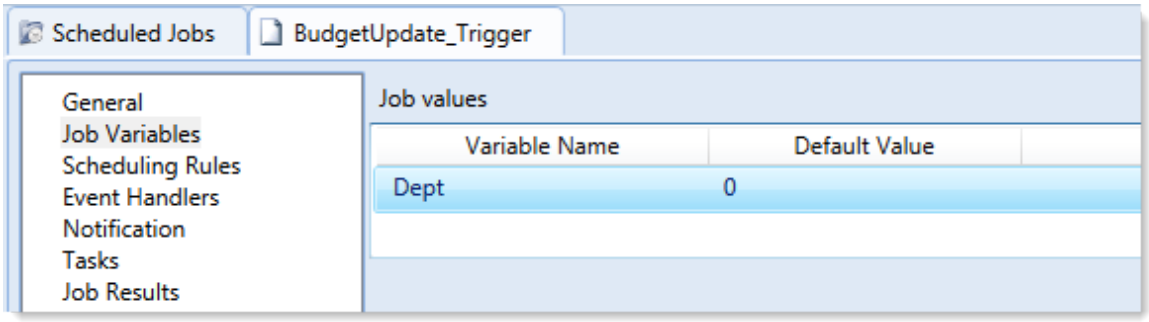
When creating the Scheduler event handler, you must determine whether to execute it as the job **Owner**, or as the job **Requester**. For the purposes of file group triggers, you most likely want to run the job as the owner. The "requester" in this case will be any user who saves a plan file in the triggering file group. That user may not have access to whatever file or file group that you want to trigger for execution in the Scheduler job. Therefore, if you want the triggered Scheduler job to always run when triggered, it should be set to the owner.

## Using variables in the job

If the file group trigger has defined variables, you can use the variables in the Scheduler job to impact the job execution in some way.

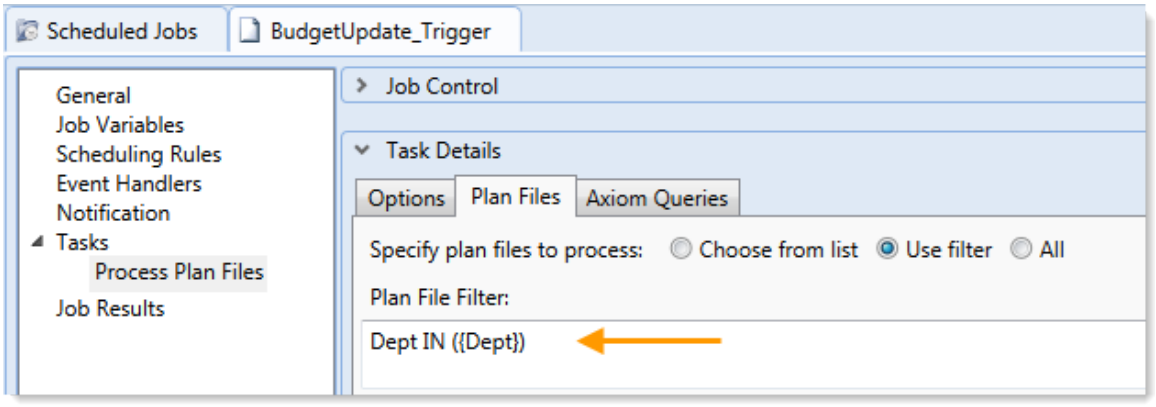
To use the trigger variables in the Scheduler job, you must:

- Define a job variable with the same name as the trigger variable. For example, if the trigger variable is named DEPT, then you must create a job variable with the same name. For example:




**NOTE:** It is recommended to define a default value for the variable. Leaving the default value blank may cause a validation error in the job settings, if blank is not a valid value where the variable is used. In this example, the default value is intentionally a non-existent department number; its sole purpose is to prevent validation errors within the Scheduler job settings.

- Use the variable in the job settings as needed to achieve the desired results. Place the variable in curly brackets as normal. For example:




- If the variable is being used in a filter (like in the example above), keep in mind the following:
    - IN syntax should be used so that the filter will be valid when the trigger variable returns multiple values.
    - If the trigger variable points to a string column, you do *not* need to place quotation marks around the variable when creating the filter. When the variable values are passed to Scheduler by the trigger, they will be automatically placed in quotation marks as necessary. Extra quotation marks will cause an error. However, keep in mind that your default variable value defined in the **Job Variables** section should have quotation marks, if you want to use the default value for testing.

In this example, we are using a file group trigger to automatically execute a Process Plan Files job for a second, dependent file group. In this particular case, both file groups use the DEPT table as the plan code table. For every department code where changed data was saved in the first file group, we want to process the plan file for that same code in the second file group. Therefore DEPT is used as a variable.

Imagine that a Process Plan Files utility was run for the first file group, and three plan files had changed data: 100, 400, and 500. Since the trigger variable is based on the DEPT column, it will pass those values to the Scheduler job as a comma-delimited string (including quotation marks as necessary). In the Scheduler job, this will cause the filter to be resolved as `DEPT IN (100,400,500)`, and therefore those three plan files will be processed for the second file group.

### Setting up email notifications for the job

When you set up the Scheduler job for the trigger, you can use the standard email notification addresses, or you can use the optional notification addresses for the triggering file group.

Using the standard email notification addresses, you can specify "hard-coded" email addresses, or you can use system variables such as {CurrentUser.EmailAddress} or {JobOwner.EmailAddress}.

The optional notification addresses for file groups can be used to provide a more granular level of notification. For example, you can define unique email address for each individual code in the plan code table if desired. Using the {NotificationAddress} system variable, you can send notifications to those file group-specific notification addresses.

# Using email notifications with triggers

File groups support optional email notification addresses that can be used to send targeted Scheduler notifications based on the file group's plan code table, rather than using the standard Scheduler notification addresses.

**NOTE:** Currently, these file group notification addresses are only for use with triggers. The Scheduler job will only be able to resolve the {NotificationAddress} variable if the job was executed in response to a file group trigger.

To define file group-specific notification addresses, you can use one or both of the following options:

- **Default address:** You can specify a default notification address for the file group.
- **Address column:** You can set up a column in the plan code table to define email addresses for each individual plan code in the file group.

If you use both options, then the default address will only be used if the column entry is blank for a particular plan code.

## Defining file group notification addresses

File group notification addresses are defined in the file group properties:

1. On the **Axiom** tab, in the **Administration** group, click **Manage > File Groups**.

   The **Axiom Explorer** dialog opens, with the focus on the **File Groups** folder.

2. Navigate to the file group that you want to edit, then right-click the file group and select **Edit File Group**.

3. In the **Edit File Group** dialog, on the **File Group** tab, complete the following settings, and then click **OK**.

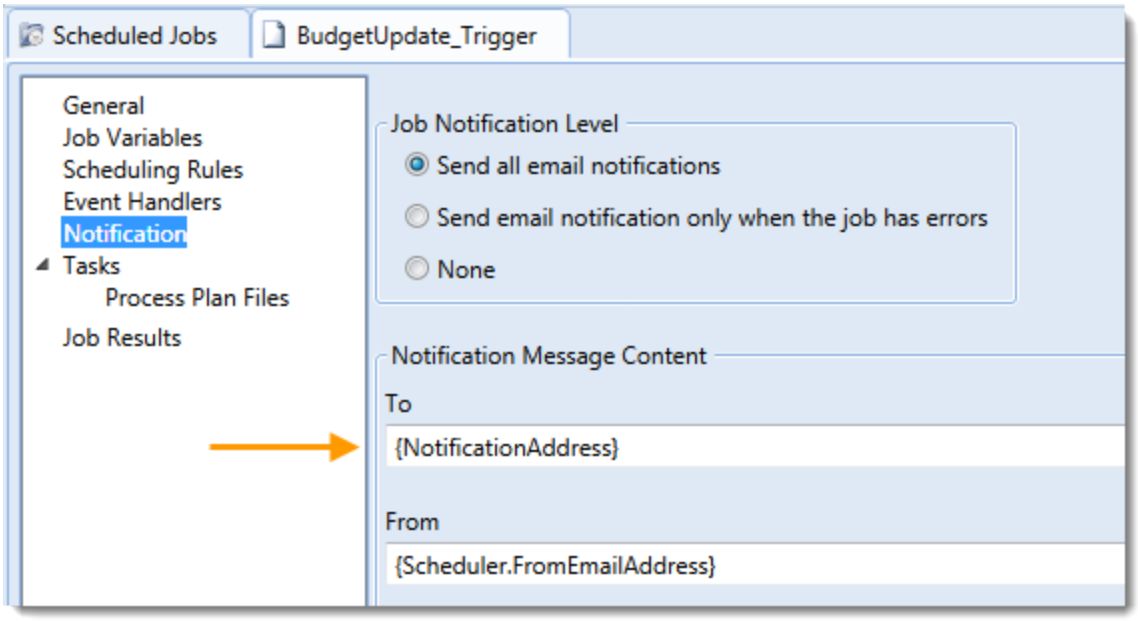
| Item | Description |
|---|---|
| Default Notification Address | The default notification email address for the file group. Separate multiple addresses with semicolons.<br><br>The default notification address is used if no Notification Address Column is specified, or if a column is specified but a plan code does not have an assigned value in the column. |

| Item | Description |
|---|---|
| Notification Address Column | The column in the plan code table that contains an email notification address for each plan code. You can select from any non-key string column in the plan code table.<br><br>Within the column, if you want to specify multiple email addresses for a plan code, separate the addresses with semicolons.<br><br>If this column is left blank for a plan code, then the default notification address is used (if defined). |

The email addresses are now available to be used within Scheduler jobs that are set up for use with file group triggers.

### Using file group notification addresses in Scheduler jobs

To use the file group notification addresses in a Scheduler job, use the system variable {NotificationAddress}. For example, you could set the To address of the job notification settings to `{NotificationAddress}`.




When the file group trigger queue is evaluated, the plan codes that triggered the event are grouped by notification address, so that there will be one Scheduler job for each unique notification address. This allows for the plan code-specific notifications held in the Notification Address Column (if used).

**Example**

Twenty plan codes are added to the trigger queue as a result of a Process Plan Files utility:

- If the notification address settings are not used, or if only the default address is used, then all twenty plan codes are evaluated as a single group. For each Scheduler job that matches the trigger event, one instance of that job is added to the Scheduler queue. All twenty codes will be considered for the evaluation of variables in that Scheduler job.
- If a table column is used for the notification address settings, and the twenty plan codes are associated with five unique email addresses, then the twenty codes are grouped into five different groups. For each Scheduler job that matches the trigger event, five instances of that job are added to the Scheduler queue. In each job, only the codes associated with that unique email address will be considered for the evaluation of variables in that Scheduler job.

If the file group trigger is being used to perform an action on a second, dependent file group, keep in mind that the notifications are based on the original "triggering" file group, not on the file group where the triggered action is being performed. For example, if file group A triggers a Process Plan Files task for file group B, the originating plan codes and therefore the corresponding notification addresses are obtained from file group A, not file group B.

# File Group Properties

The following properties are available for file groups and file group scenarios. These properties are managed in Axiom Explorer (**Administration > Manage > File Groups**). To access these properties, right-click a file group and select **Edit**.

File group properties can be referenced in Axiom files by using the following functions: GetFileGroupID, GetFileGroupProperty, and GetFileGroupVariable.

## File Group tab

This tab defines basic properties for the file group.

### General Properties

| Item | Description |
|---|---|
| File Group Name / Scenario Name | The name of the file group or the file group scenario. You can use any file group variable in the name, including built-in variables such as `{FileGroupYear}`.<br><br>If the current file group is a scenario, then the name cannot be edited.<br><br>**NOTES:**<br>• File group names cannot contain characters that are not allowed in Windows file paths. This includes the following characters: backslash and forward slash, colon, asterisk, question mark, double quotation mark, greater than and less than, and pipe.<br>• The resolved file group name (after substituting any variables with their values) must be unique. For example, you can have two file groups where the name is defined as **Budget {FileGroupYear}**, but only if the year values are unique for each file group. If both file groups are set to the same year, then the resolved names are not unique and you will be prevented from saving the duplicate name. |

| Item | Description |
|---|---|
| File Group Year | Optional. The year of planning for the file group. For example, if the file group is being used to create the budget for the 2022 fiscal year, then the File Group Year should be set to 2022.<br><br>If defined, this value can be used in the file group name, the display name, the tab prefix, and in other file group variables. Enter the full four-digit year number, not a two-digit year number. The two-digit version can be returned by using built-in variables for the file group. For more information, see Using years as variables in file groups.<br><br>To use the file group year in the file group name, click the **use in file group name** link to the right of the box. This link only displays if a value has been defined for the year. Clicking this link adds the `{FileGroupYear}` variable to the end of the File Group Name (for example: "Budget 2022" where 2022 is the file group year). Alternatively you can type the variable within the file group name or in any other supported field. |
| File Group Category | Optional. The category for the file group. File groups that belong to a category can display together on the ribbon and in the Explorer task pane.<br><br>The category must already be defined in order to assign a file group to it. For more information, see Using file group categories. |
| Plan Code Table | The reference table that defines the list of plan codes for the file group.<br><br>For example, budgets might be planned by department, forecasts might be planned by division, and capital plans might be planned by project. Generally, each item in the specified table will have a plan file in the file group. For more information, see Plan code tables for file groups.<br><br>This setting cannot be changed once any plan files have been created for the file group.<br><br>If the selected table uses an identity column for the key column, then the file group is automatically flagged as an on-demand file group. Tables that use an identity column are indicated in the drop-down list with a special icon ( ).<br><br>**NOTES:**<br>• Key columns in plan code tables cannot contain characters that are not allowed in Windows file paths. This includes the following characters: backslash and forward slash, colon, asterisk, question mark, double quotation mark, greater than and less than, and pipe. The table will be validated for these characters when creating the file group, and when saving data to the table.<br>• Picklist tables and KPI tables are not eligible to be plan code tables. |
| Description | The description of the file group. |

Display Properties

| Item | Description |
|---|---|
| Display Name | The name to be displayed on the file group's button on the Axiom ribbon tab, in the Explorer task pane, and in other areas where file groups are displayed. This name is how users select file groups to work with.<br><br>By default, the display name is the same as the file group name. If you update the file group name, the display name will remain in sync. However, if you want to define a different display name, then type in any value. You can use any file group variable in the name, including built-in variables such as `{FileGroupYear}`.<br><br>If you have defined a different display name and you want to go back to using the file group name, click the **Use file group name** button to the right of the box.<br><br>If the current file group is a scenario, then the display name cannot be edited. |
| Tab Prefix | Optional. The text to be displayed on the file tab of files associated with this file group. The text displays before any other text on the file tab. The tab prefix applies to plan files and driver files.<br><br>The intent of this setting is to help users identify which files belong to which file groups, if they have multiple files open. For example, the tab prefix for the 2021 Budget file group could be "[BGT21]".<br><br>You can use any file group variable in the name, including built-in variables such as `{ShortFileGroupYear}`.<br><br>If left blank, then no file group identifier displays on the file tab when the file is opened. However, you can hover your cursor over the tab to see the file group name displayed in the tooltip. |

| Item | Description |
| --- | --- |
| Tab Column | Optional. Specifies an alternate column value to display in the file tab of plan files associated with this file group. The column value displays after the tab prefix (if defined). This setting also determines what displays on the progress bar when a user opens a plan file.<br><br>By default, this property is set to **None**, which means the key column value from the plan code table displays on the tab. For example, if the plan code table is Dept, then the department code displays on the tab.<br><br>If desired, you can specify an alternate column to use, such as the Description column. If a column is specified, then that column value displays on the file tab instead of the key column value. The alternate column must be a string column.<br><br>**NOTE:** Use caution before specifying any column other than the key column. The specified column should contain unique values for all plan codes, otherwise multiple open plan files may display with the same text on the file tab.<br><br>This option only applies to plan files in the file group. For other files in the file group, the file name displays on the file tab. |

If the current file group is a scenario, then the source file group for the scenario is listed at the bottom of this tab, for information purposes.

## Options tab

This tab defines various options for the file group.

### Template options

| Item | Description |
|---|---|
| Allow Generation of Plan Files from Templates | Select this check box to allow use of the **Create Plan Files** utility for the file group, to create plan files from templates.<br><br>This option is selected by default for standard file groups, and disabled by default for on-demand file groups. However, you can change the setting for either file group as desired.<br><br>• If this check box is selected, then the **Create Plan Files** utility is available for the file group, for users with the appropriate rights.<br>• If this check box is not selected, then the utility is not available for the file group. For on-demand file groups, this is the default state, and users can create plan files as normal using the Open Plan Files dialog.<br><br>You might disable this check box for a standard file group as a means of "freezing" plan file creation for the file group, or if the file group is intended to be used only as a target file group for copying plan files. |
| Default Template | The default template to use when building plan files for the file group. You can type a template name, or select from a drop-down list of existing templates for the file group. Do not include the file extension on the name.<br><br>File group variables can also be used in this setting. You might do this in order to dynamically change the default template by using a Save Type 4 utility that changes the value of the variable.<br><br>This setting is used in the following ways:<br>• If no template column is specified, then the default template is used to create all plan files.<br>• If a template column is specified, but a particular plan code does not have an entry in that column, then the default template is used for that plan code. However, if a plan code has an entry in the template column, but the entry is invalid, the default template is not used and an error results.<br><br>**NOTE:** If you are creating a new file group, then no templates exist for that file group yet. You can either manually type the name that you intend to give a template, or you can modify the settings later after you have created templates. |

| Item | Description |
| --- | --- |
| Template Column | The column in the plan code table that contains the template assignment for each plan code. You can select from any non-key string column in the plan code table.<br><br>You have the following options to complete the template settings for a file group:<br>• You can define a template column and a default template. In this case, the default template is only used if a plan code does not have an entry in the template column.<br>• You can define only a template column. In this case, an error results if a plan code does not have an entry in the template column.<br>• You can define only a default template. In this case, the default template is used for all plan codes.<br><br>For more information, see Assigning templates to plan codes.<br><br>**NOTE:** For on-demand file groups, special considerations apply if you want to use a template column. For more information, see Template options for on-demand file groups. |

### Plan File Options

| Item | Description |
| --- | --- |
| Enable Plan File Attachments | Optional. Select this check box to enable storing file attachments as supporting documents for plan files.<br><br>This option is disabled by default, which means that file attachment features will not be available for the file group.<br><br>If this option is enabled, then file attachment features will be available for the file group. Users can add, delete, and view file attachments that are associated with a particular plan file, depending on their security permissions to the plan file. For more information, see Using file attachments with file groups. |
| Use Virtual Plan Files | Optional. Select this check box to use virtual plan files with the file group.<br><br>This option is disabled by default, which means that persistent, physical plan files are generated and stored for each plan code in the file group.<br><br>If this option is enabled, then physical plan files are not stored for the file group. Instead, temporary, virtual plan files are generated as needed from the template. The template must be designed as "rebuildable" in order to support virtual plan files. For more information on using virtual plan files, see Using virtual plan files. |

| Item | Description |
| --- | --- |
| Process Plan Files with Utilities | Optional. Select this check box if the default processing mode for plan files should be to process with utilities instead of normal processing. This is primarily intended for file groups that use form-enabled plan files, where the content of each plan file is sourced from various utility files instead of defined within the plan file itself (by use of embedded forms).<br><br>If enabled, this setting impacts the following behavior for this file group:<br>• The default processing mode for Process Plan Files is now **Process with Utilities** instead of **Normal Processing**.<br>• If a step in a plan file process definition is set to **Save and validate plan file before advancing to next step**, utility processing is performed for the plan file instead of saving the plan file itself. |
| Use Composite Plan Files | If this option is enabled, plan files in the file group consist of multiple files—the main plan file and then one or more *plan file parts*. After opening the main plan file, you can open plan file parts using either an associated task pane or a GetDocument function. The plan file parts are considered to be part of the plan file and display as a group.<br><br>Composite plan files use multiple templates to create the main plan file and the parts. When plan files are created, the primary assigned template is used to create the main plan file. When the main plan file is processed, the associated plan file parts are dynamically created as needed and processed based on a CompositePlanFiles data source defined in the main plan file.<br><br>This feature can only be used with spreadsheet plan files. Composite plan files can only be opened in the Windows Client.<br><br>**NOTES:**<br>• This option can only be enabled and configured by product developers, to be used with product-delivered file groups. Otherwise, the option does not display. For more information on how to work with a file group when this feature is enabled, consult the appropriate product documentation. Syntellis employees should consult internal resources for more information on this feature as needed.<br>• It is also possible to create form-enabled plan files that consist of multiple files, using the Embedded Form component. The composite plan files feature does not apply to this use case. |

| Item | Description |
| --- | --- |
| Show On List Column | Optional. The column in the plan code table that specifies whether a plan code should be included in file group lists and available to file group processes. You can select from any Boolean column in the plan code table.<br><br>If no column is specified, then all plan codes will be included in file group lists and available to file group processes.<br><br>If a column is specified, and a plan code is set to **False** in that column, then that plan code will not display in dialogs that list plan files, such as **Create Plan Files** or **Open Plan Files**. The plan code cannot be included in file group processes.<br><br>For more information, see Using a Show On List column. |
| Disable Clone Existing Plan Files | Optional. Select this check box if you do not want to allow users to create new plan files by cloning existing plan files. By default this option is not selected, which means that the "clone existing" feature is enabled and available.<br><br>If this option is selected, then the **Clone selected item** command is hidden in the Open Plan Files dialog.<br><br>**NOTE:** This option is only available for on-demand file groups. Standard file groups do not allow creating new plan files by cloning existing plan files. |

### Notification Options

These options only apply if you are using file group triggers. For more information see Using email notifications with triggers.

| Item | Description |
| --- | --- |
| Default Notification Address | Optional. The default notification email address(es) for the file group. Separate multiple addresses with semicolons.<br><br>The default notification address is used if no Notification Address Column is specified, or if a column is specified but a plan code does not have an assigned value in the column.<br><br>This address can be used in Scheduler jobs by using the system variable {NotificationAddress}. A value will only be passed when the Scheduler job is executed by use of file group triggers. |

| Item | Description |
|---|---|
| Notification Address Column | Optional. The column in the plan code table that specifies a notification email address for each plan code. You can select any non-key string column in the plan code table.<br><br>If this column is left blank for a plan code, then the default notification address is used (if defined).<br><br>Within the column, if you want to specify multiple email addresses for a plan code, separate the addresses with semicolons.<br><br>This address can be used in Scheduler jobs by using the system variable {NotificationAddress}. A value will only be passed when the Scheduler job is executed by use of file group triggers. |

### Process Options

| Item | Description |
|---|---|
| Plan File Process | Optional. Specify a plan file process for the file group. You can select any plan file process definition located in the Processes folder for the file group. Standard process definitions cannot be used.<br><br>For more information, see the *Plan File Process Guide*. |

### On Demand Options

These options are only available if the plan code table for the file group uses an identity column as its key column. The use of the identity column enables the ability to automatically generate a sequential plan code "on demand." For more information, see On-Demand File Groups.

| Item | Description |
|---|---|
| Add File Message | Enter the text that you want to display to users to create a new plan file, such as "Add new file" or "Add new capital request".<br><br>This text will display in the top right-hand corner of the Open Plan Files dialog in the Excel Client and the Windows Client.<br><br>This text is required, even if end users will not be using this option to create plan files (for example, if all end users will use the Web Client). |

| Item | Description |
| --- | --- |
| Add File Form | Optional. Specify an Axiom form to use as the "input form" to collect starting values for the plan code table when creating a new plan file. You can select any form-enabled file stored in the Utilities folder for the file group (recommended), or in the Reports Library.<br><br>If an Axiom form is specified, then when a user clicks the "add new file" command in the Open Plan Files dialog, the form will be opened as a "dialog" within the application so that the user can complete inputs on the form. This form replaces the built-in prompt for validated columns.<br><br>For more information on using this feature, see Using an Axiom form as an "add file" dialog. |
| Clone File Form | Optional. Specify an Axiom form to use as the "input form" to collect starting values for the plan code table when creating a new plan file by cloning an existing plan file. You can select any form-enabled file stored in the Utilities folder for the file group (recommended), or in the Reports Library.<br><br>This form is used when a user clicks the **Clone selected item** option for the on-demand file group within the Axiom Excel Client or Windows Client. For more information on using this option, see Using an Axiom form as an "add file" dialog.<br><br>**NOTE:** If **Disable Clone Existing Plan Files** is selected (in the **Plan File Options** group), then the Clone File Form option is disabled because users cannot create plan files by cloning existing plan files. |

## Variables tab

Each file group can have a set of defined variables that are then referenced in file group components such as templates, drivers, and utilities. These variables are used to define key file group properties such as important planning assumptions and the tables used by the file group.

Use of variables can help simplify file setup within the file group, and can streamline the process of cloning and reconfiguring a file group for an "annual rollover" or for scenario planning. For more information, see File Group Variables.

The Variables tab is split into three different sub-tabs:

- General Variables: Defines variables for general file group information and assumptions.
- Table Variables: Defines variables for the tables used by the file group.
- Picklist variables: Defines variables for the picklist tables used by the file group.

## Display Columns tab

You can configure the display columns for the file group. These columns are displayed in dialogs that show lists of plan files for the file group.

These settings define the columns to show in dialogs, as well as various display attributes for those columns. You can also specify which columns to use for sorting and grouping.

The Display Columns tab is split into two sub-tabs:

- Plan File Columns: Defines display columns for file group dialogs such as Open Plan Files and Process Plan Files.
- Process Columns: Defines display columns for the Process Status dialog, when viewing plan file processes for this file group. For more information, see the *Plan File Process Guide*.

## Web Configuration tab

You can configure display properties for the associated web pages for the file group in the Axiom Web Client. These web pages are typically used when plan files are form-enabled and the primary client for end users is the Web Client.

The configurable properties include the page header text, the columns to show and the sort order, and display properties for each column. You can also optionally specify a workbook that defines refresh variables to allow filtering of the page.

The Web Configuration tab is split into two sub-tabs:

- Plan File Directory: Defines display properties for the Plan File Directory page.
- Process Directory: Defines display properties for the Process Directory page. For more information, see the *Plan File Process Guide*.

## Triggers tab

Using file group triggers, you can "trigger" the execution of a Scheduler job after data is saved from plan files in the file group, based on a specified event. For more information, see File Group Triggers.

To add a trigger event, click **Add Event**. To delete a trigger event, right-click the event and select **Remove Event**. Each event has the following settings:

| Item | Description |
| --- | --- |
| Enabled | Specifies whether the event is active. This check box is enabled by default, which means that the event will be triggered any time data is saved from plan files in the file group.<br><br>To disable the event, clear the check box. In this case, saving data will not trigger the event. This is primarily intended for situations where you want to temporarily disable the event, with the intent of re-enabling it at some future time.<br><br>If you no longer need an event, you can delete it instead of disabling it. |
| Event Name | The name of the event to trigger when data is saved from plan files in the file group.<br><br>When the event is triggered, Axiom looks for any Scheduler jobs that contain an event handler with this name. If any jobs are found, they are added to the schedule and are eligible to be processed immediately, depending on Scheduler thread availability and any other higher-priority jobs already in the queue.<br><br>If no matching jobs are found, no further action occurs, and no errors are generated. |
| Variables | Optional. One or more variables to associate with the event. These variables can be used within the triggered Scheduler job.<br><br>To create a variable, click **Add Variable**. To delete a variable, right-click the variable and select **Remove Variable**.<br><br>Each variable has the following properties:<br><br>• **Variable Name**: The name of the variable. To use the variable in the triggered Scheduler job, create a corresponding job variable with the same name and then use the variable in the desired task settings.<br>• **Column**: A column in the plan code table that contains the values for the variable. Multiple values will be passed as a comma-delimited list.<br>• **Value**: A fixed value for the variable.<br><br>You can use the column or the value to define the variable value. |


# Save Type 4 for File Groups

Save Type 4 can be used to modify certain file group properties from within a spreadsheet. This allows you to change properties based on queries, calculations, and inputs in a spreadsheet, rather than using the software interface. Additionally, Save Type 4 utilities can be scheduled for processing using Scheduler's Process Document List task.

## Managing file group aliases using Save Type 4

Using Save Type 4, you can create, edit, or delete file group aliases by using save-to-database within a spreadsheet, instead of using Axiom Explorer. This may be a more convenient approach when you want to impact many aliases at one time, or when you want to automate the process.

Save Type 4 depends on the placement of save-to-database tags within the sheet. There are three components:

- The primary SaveStructure2DB tag, which defines the locations of the save-to-database control row and control column, and specifies the system table that you want to update.
- Column tags in the save-to-database control row, to specify the columns holding the alias properties.
- Row tags in the save-to-database control column, to specify the rows to include in the save process and to determine the action to take on the alias.

Save-to-database tag summary

| Tag Type | Tag Syntax |
|---|---|
| Primary tag | `[SaveStructure2DB; Axiom.FileGroupAliases; CustomSaveTag=`*`Name`*`]` |
| Row tags | `[Save]`<br>`[Delete]` |
| Column tags | `Alias`<br>`FileGroupName`<br>`Description` |

**NOTES:**

- Save Type 4 must be enabled for the sheet on the file's Control Sheet in order for the save process to occur.
- The user performing the save must be an administrator or have the **Administer File Groups** security permission.

### Placing the primary save-to-database tag in the sheet

To define the save-to-database process, place the following tag in any cell in the sheet, within the first 500 rows:

```
[SaveStructure2DB;Axiom.FileGroupAliases]
```

The row containing this tag becomes the control row for the process, and the column containing this tag becomes the control column for the process.

You can also optionally use the custom save tag parameter. For example:

```
[SaveStructure2DB;Axiom.FileGroupAliases;CustomSaveTag=SaveAlias]
```

**NOTES:**

- The primary SaveStructure2DB tag must be located in the first 500 rows of the sheet.
- The SaveStructure2DB tag can be placed within a formula, as long as the starting bracket and identifying tag are present as a whole within the formula.

### Placing the alias property tags in the control row

Within the control row for the save-to-database process, specify the columns that define the variable properties. These properties can be placed in any column, to either the right or the left of the SaveStructure2DB tag.

| Column Tag | Description |
|---|---|
| `Alias` | The name of the file group alias. |
| `FileGroupName` | The name of the file group. Use the real file group name (not the display name) with all variables resolved to their current values. For example, if the file group name is `Budget {CurrentFileGroupYear}`, then use `Budget 2022` (where 2022 is the current file group year).<br><br>The specified file group must already exist. |
| `Description` | The description of the alias. |

Alias and FileGroupName are required to create or edit an alias.

### Indicating the rows to save or delete

Within the control column for the save-to-database process, mark each row that you want to be processed with the appropriate row tag:

| Row Tag | Description |
|---|---|
| `[Save]` | Creates or updates the alias with the properties specified in the spreadsheet.<br><br>**NOTE:** If you have defined a custom save tag in the SaveStructure2DB tag, then you must mark the rows with that tag instead of the default tag. For example, if your primary tag is `[SaveStructure2DB; Axiom.FileGroupAliases; CustomSaveTag=SaveAlias]` then you would place the tag `[SaveAlias]` in the rows that you wanted to be saved. |
| `[Delete]` | Deletes the alias, based on the specified alias name. |

Only rows that are marked with a valid tag are processed; all other rows are ignored, even if there is content in the property columns. If a row contains a valid tag but no content exists in the property columns, a save error will occur.

When the save-to-database occurs, delete actions are processed first, followed by save actions.

**NOTE:** The row tag can be placed within a formula if desired. For example, you might want to use a formula to determine whether a particular row should be saved or deleted.

### Populating the alias properties in the spreadsheet

You can manually type the alias properties within the spreadsheet, or you can use an Axiom query to populate the spreadsheet with the aliases that are currently defined in the database. Once you have the current list of aliases as a starting point, you can adjust them as necessary, and then save the changes back to the database.

For example, imagine that you want to update aliases for a rollover. You can:

- Create an Axiom query to bring in the current file group aliases, by querying Axiom.FileGroupAliases.
- Set up the save-to-database tags so that the fields to be saved line up with the information brought in by the Axiom query.
- Edit the FileGroupName property for the aliases so that the aliases now point to the appropriate file groups. For example, if the Current Budget alias currently points to Budget 2021, you can update it to point to Budget 2022.
- For the aliases that you edited, place a `[Save]` tag in the save-to-database control column.

When you perform the save-to-database, the aliases will be updated to point to the different file groups.

# Managing file group variables using Save Type 4

Using Save Type 4, you can create, edit, or delete file group variables by using save-to-database within a spreadsheet, instead of using the **Edit File Group** dialog. This may be a more convenient approach when you want to impact many variables at one time, or when you want to automate the process.

Save Type 4 depends on the placement of save-to-database tags within the sheet. There are three components:

- The primary SaveStructure2DB tag, which defines the locations of the save-to-database control row and control column, and specifies the system table that you want to update.
- Column tags in the save-to-database control row, to specify the columns holding the variable properties.
- Row tags in the save-to-database control column, to specify the rows to include in the save process and to determine the action to take on the variable.

Save-to-database tag summary

| Tag Type | Tag Syntax |
|---|---|
| Primary tag | `[SaveStructure2DB; Axiom.FileGroupVariables; CustomSaveTag=`*`Name`*`]` |
| Row tags | `[Save]`<br>`[Delete]` |
| Column tags | `FileGroupID`<br>`Name`<br>`RawValue`<br>`Category`<br>`ReadOnly`<br>`DataType` |

**NOTES:**

- Save Type 4 must be enabled for the sheet on the file's Control Sheet in order for the save process to occur.
- The user performing the save must be an administrator or have one of the following security permissions: **Administer File Groups** (for all file groups) or **Modify File Group** (for specific file groups).
- The built-in file group variables such as FileGroupYear cannot be modified using Save Type 4.

### Placing the primary save-to-database tag in the sheet

To define the save-to-database process, place the following tag in any cell in the sheet, within the first 500 rows:

```
[SaveStructure2DB;Axiom.FileGroupVariables]
```

The row containing this tag becomes the control row for the process, and the column containing this tag becomes the control column for the process.

You can also optionally use the custom save tag parameter. For example:

```
[SaveStructure2DB;Axiom.FileGroupVariables;CustomSaveTag=SaveVariable]
```

**NOTES:**

- The primary SaveStructure2DB tag must be located in the first 500 rows of the sheet.
- The SaveStructure2DB tag can be placed within a formula, as long as the starting bracket and identifying tag are present as a whole within the formula.

### Placing the variable property tags in the control row

Within the control row for the save-to-database process, specify the columns that define the variable properties. These properties can be placed in any column:

| Column Tag | Description |
|---|---|
| `FileGroupID` | The database ID of the file group. |
| `Name` | The name of the variable. |
| `RawValue` | The variable value. |
| `Category` | The variable category. Use Generic, TableName, or Picklist. "Generic" refers to non-table variables defined on the General Variables tab. |
| `ReadOnly` | Whether the table the variable resolves to is read-only for the file group (True/False). Only applies if the category is TableName. |
| `DataType` | The variable data type. Use either String or Numeric. This field is only configurable for generic (general) variables. All other variable types are always String.<br><br>When editing an existing general variable, you can change a Numeric variable to String, but you cannot change a String variable to Numeric. |

The column tags can be placed to either the right or the left of the SaveStructure2DB tag. FileGroupID, Name, and Category are required.

The control row must be dedicated to containing only valid column names for the Save Type 4 operation to the target table. Any invalid entries in the control row will cause an error when saving.

**NOTES:**

- It is not possible to configure the scenario cloning behavior of a table variable using Save Type 4. If a new variable is created, it uses the default behavior (Automatic).
- It is not possible to configure the optional properties of picklist variables using Save Type 4. These properties can only be modified in the **Edit File Group** dialog.

### Flagging the rows to save or delete

Within the control column for the save-to-database process, mark each row that you want to be processed with the appropriate row tag:

| Row Tag | Description |
|---|---|
| `[Save]` | Creates or updates the variable with the properties specified in the spreadsheet.<br><br>**NOTE:** If you have defined a custom save tag in the SaveStructure2DB tag, then you must mark the rows with that tag instead of the default tag. For example, if your primary tag is `[SaveStructure2DB; Axiom.FileGroupVariables; CustomSaveTag=SaveVariable]` then you would place the tag `[SaveVariable]` in the rows that you wanted to be saved. |
| `[Delete]` | Deletes the variable, based on the specified name. |

Only rows that are marked with a valid tag are processed; all other rows are ignored, even if there is content in the property columns. If a row contains a valid tag but no content exists in the property columns, a save error will occur.

When the save-to-database occurs, delete actions are processed first, followed by save actions.

**NOTE:** The row tag can be placed within a formula if desired. For example, you might want to use a formula to determine whether a particular row should be saved or deleted.

### Populating the variable properties in the spreadsheet

You can manually type the variable properties within the spreadsheet, or you can use an Axiom query to populate the spreadsheet with the variables that are currently defined in the database. Once you have the current list of variables as a starting point, you can adjust them as necessary, and then save the changes back to the database.

For example, imagine that you want to update variables for the 2022 Budget file group. You can:

- Create an Axiom query to bring in the current variables for that file group, by querying Axiom.FileGroupVariables and filtering the results to just that file group.

- Set up the save-to-database tags so that the fields to be saved line up with the information brought in by the Axiom query.
- If you want to edit the properties of existing variables, you can change the properties and then place a [Save] tag in the save-to-database control column. If you want to delete an existing variable, you would use a [Delete] tag.
- If you want to create a new variable for that file group, you can enter the variable properties, then place a [Save] tag in the save-to-database control column.

When you perform the save-to-database, the variables will be created, updated, or deleted as appropriate.

# Index

## P



## R



## S



## T

## U



## V