What's New
Axiom Software
Version 2019.3

KaufmanHall
AXIOM

Contents

Introduction

Kaufman Hall is pleased to announce the release of **Axiom Software Version 2019.3**. This release features enhancements to various areas of the software, such as:

- **File Collect**: Perform file collect using multipass processing, to dynamically iterate file collection and create report packages based on a specified dimension. Additionally, a new feature is available to consolidate multiple attachments into a single email.
- **File Groups**: New commands to open the Plan File Directory or Process Directory for a specified file group, to allow launching the web directories from task panes, ribbon tabs, and Axiom forms.
- **Imports**: Ability to import data into a table with an identity column, and either auto-generate new identity records or create records with specific identity values.
- **Performance optimizations**: Various enhancements can be leveraged to improve file performance, including the ability to batch Axiom queries across sheets, process multiple save-to-database blocks concurrently, and use a lightweight method to return related values in plan files.

This *What's New* document provides information on all new features and enhancements in this release. Reviewing this document should give you a basic understanding of how these new features work, and what benefits they may provide to your organization. For full details on any new feature, please see the Axiom Software Help files or the PDF guides.

IMPORTANT: Before upgrading to version 2019.3, make sure you have reviewed the separate *Release Notes* document to understand any important technical changes and upgrade considerations in this release.

File Processing

Multipass processing for file collect

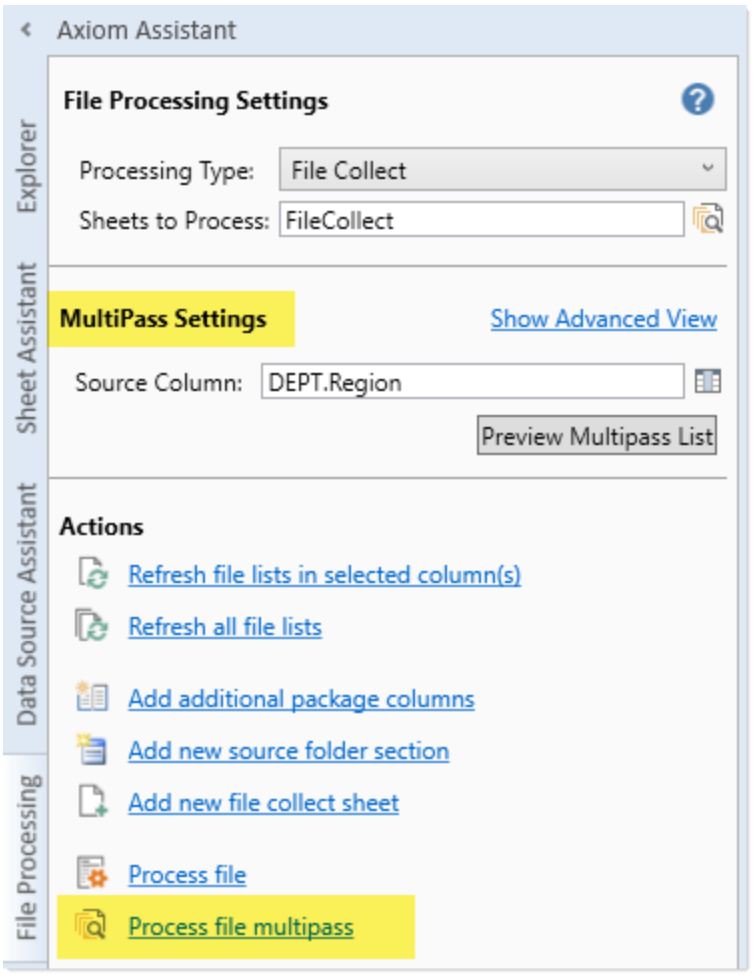
File collect can now use multipass processing to iteratively generate report packages over a designated dimension, such as by Facility, Region, or VP. This can streamline the setup and maintenance necessary to perform file collect. Instead of needing to define multiple file collect packages for different recipients, in many cases you can now define one file collect package and use multipass processing to dynamically adjust the package settings per pass.

For example, imagine that your organization has six regions and you want to create and deliver a report package for each region. When using non-multipass processing, this requires you to define six different package columns in the File Collect Configuration Sheet—one for each region. Each package column needs to use different settings such as email addresses, output file names and/or locations, and file filters in order to generate a unique report package for each region.

When using multipass processing, you can define just one package column in the File Collect Configuration sheet. If the source column for multipass processing is Region, then this single package column will be evaluated six times, using a different region for each pass. Assuming that the package column uses dynamic settings, then each pass in multipass processing is resolved differently to generate a unique report package for each region. The function GetCurrentValue can be used in the file collect settings to dynamically return information about the current pass, and a special multipass filter template can be used to dynamically generate the list of files to collect.

Multipass settings

When a file is enabled for file processing and uses **File Collect** as the **Processing Type**, the **MultiPass Settings** are now available in the File Processing task pane. Additionally, the ability to **Process file multipass** is now available in the task pane and from the **File Processing** menu on the ribbon.

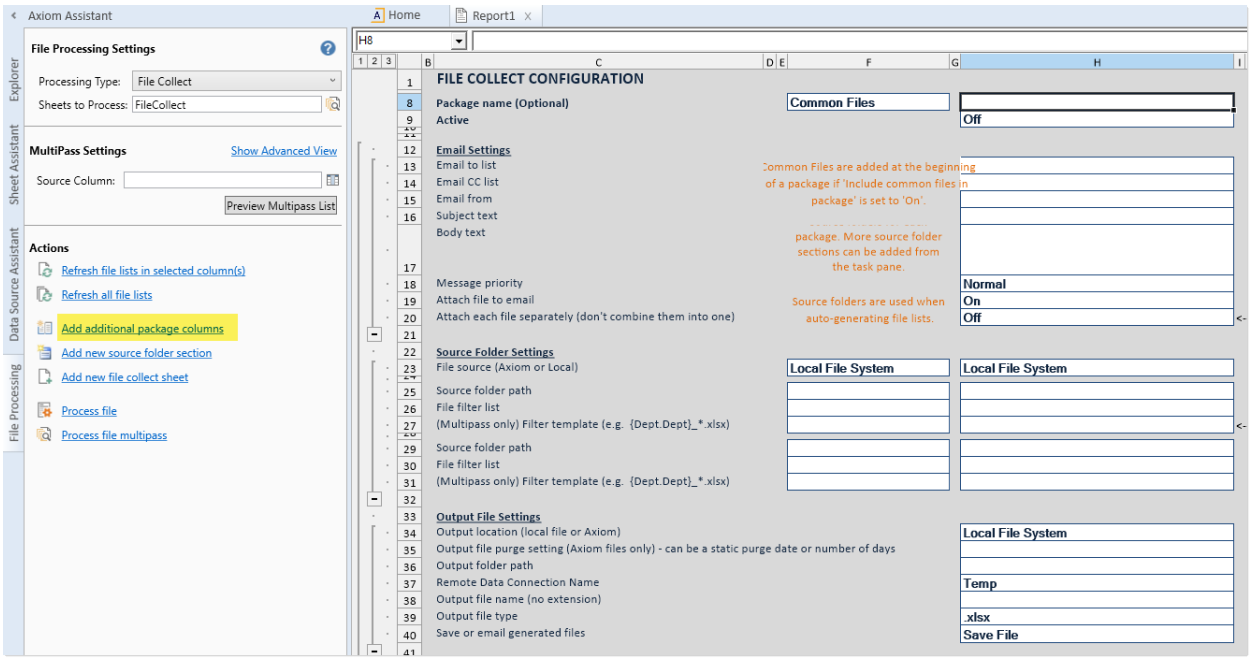


New multipass options available for file collect processing

If you define multipass settings and then perform file collect processing using multipass, the file collect configuration sheet is iteratively processed over the values in the specified multipass column. All enabled file collect package columns will be processed once per pass.

This works in a similar manner as normal multipass processing, but the goal is slightly different. For example, when performing snapshot multipass processing on a report, you are filtering the report data per pass. When performing multipass processing for file collect, the goal is not to filter report data but simply to dynamically change the package settings per pass.

The standard File Collect Configuration Sheet now only has one package column by default (plus the common files column), to encourage use of multipass processing whenever it is appropriate. This applies whenever you add a new File Collect Configuration Sheet to a file. However, you can still add more package columns as needed using the **Add additional package columns** action in the File Processing task pane. You can add more package columns when you cannot use multipass processing, or when you are using multipass processing but you want to process multiple packages per pass.



Example new File Collect Configuration Sheet with single package column

Using GetCurrentValue in file collect settings

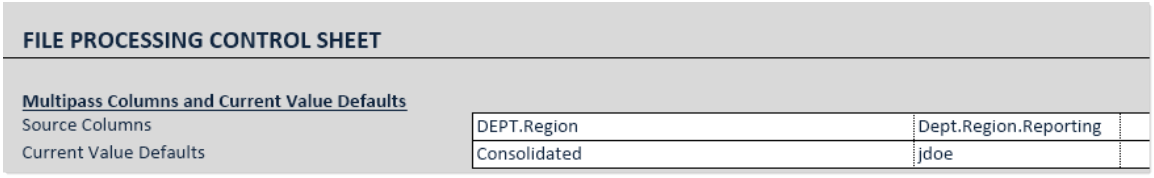
You can use the GetCurrentValue function in various file collect settings to dynamically change the settings per pass. The most common use case is in the output file name and/or folder path, and in the email settings to determine the recipient's email address. You might also use the function in the source folder path, if the source files for each path were saved to unique folder locations.

The two most common ways to use the GetCurrentValue function are as follows:

- The syntax `GetCurrentValue()` returns the name of the current pass value. If you are multipass processing by Dept.Region, and the current pass is for region US West, the function returns "US West".

- The syntax `GetCurrentValue("ColumnName")` returns the value in that column for the current pass. For example, the column Dept.Region looks up to the Region table, and the Region table contains a column named Reporting that holds the target user to receive the report package for each region. If the current pass is for region US West, the function `GetCurrentValue ("Dept.Region.Reporting")` returns the name of the user to receive the report package for that region.

  In order for GetCurrentValue to return a value for a particular column, that column must be used in the multipass settings, typically as a secondary source column. To add more source columns, you can use the advanced multipass settings in the File Processing task pane, or you can go to the File Processing Control Sheet (Control_FileProcessing) and add the additional source columns to the multipass section at the top of the sheet. When editing the File Processing Control Sheet directly, you can also specify default values that the GetCurrentValue function will return when multipass processing is not occurring.



Example source columns and default values on File Processing Control Sheet

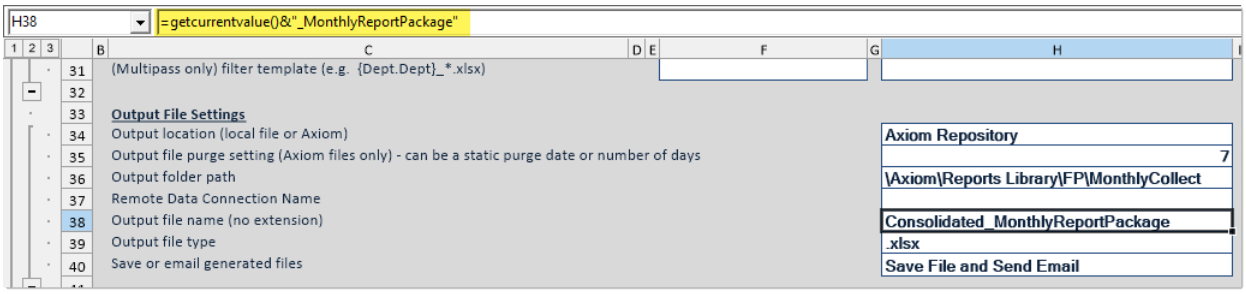
For more information, see the following topics in Axiom Software Help:

- *GetCurrentValue function* (AX2354)
- *Returning related values for each pass* (AX1586)

NOTE: It is *not* possible to use the file processing variables such as `[Current_Value]` in the File Collect Configuration Sheet. File processing variables are only valid for use in the File Processing Control Sheet.

Output file name and/or folder

The following example shows GetCurrentValue being used to set the output file name, so that the file name changes for each pass. For example, when the pass is performed for region "US West", the name of the file will be resolved as "US West_MonthlyReportPackage". The function returns "Consolidated" when multipass processing is not occurring, because that is the current value default set on the File Processing Control Sheet (as seen on the previous screenshot).

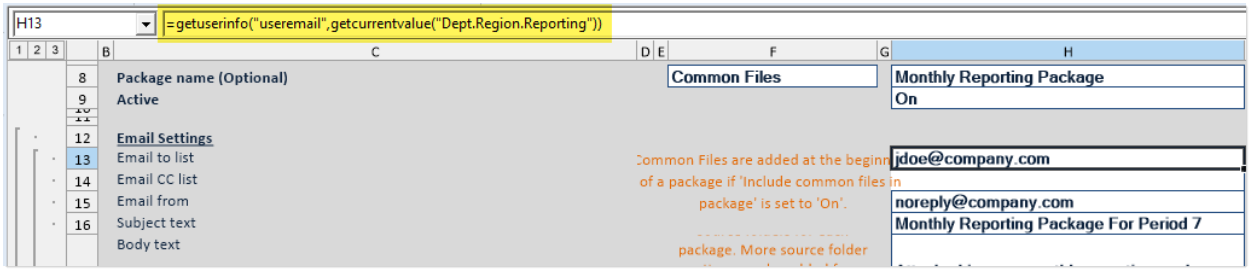


Example output file name set using GetCurrentValue

The same concept could be used to change the output folder path per pass, if desired.

Email recipients

The following example shows GetCurrentValue being used to return the name of the user who should receive the report package for each region. In this example, the column Dept.Region looks up to the Region table, and the Region table contains a column named Reporting that holds the target user. If the current pass is for region US West, the function `GetCurrentValue("Dept.Region.Reporting")` returns the name of the user to receive the report package for that region. The GetUserInfo function is then used to look up the user's email address from security.



Example email address set using GetCurrentValue

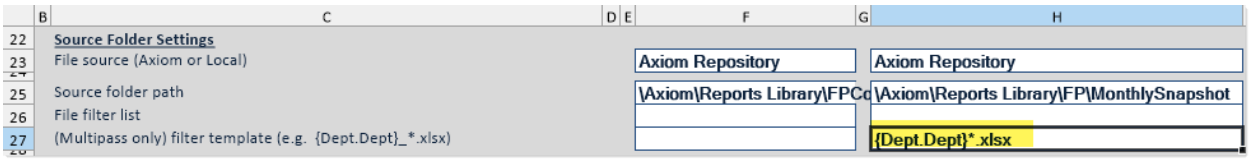
The GetCurrentValue function returns user "jdoe" when multipass processing is not occurring, because that is the current value default set on the File Processing Control Sheet.

If the target email addresses were stored in the table directly (instead of looking them up from security), then GetCurrentValue could be used on its own to return the email address.

Using a file filter template

When performing multipass processing with file collect, you can use the new **(Multipass only) Filter template** setting to dynamically filter the files to collect per pass. Although you could use GetCurrentValue in the file filter, this new setting is a more flexible option to set the file filter for multipass processing. The multipass filter template can use a table.column name that will be resolved during each pass to dynamically set the file filter.

For example, imagine that you are performing multipass file collect processing by region, and you want to collect all files that contain a department code that belongs to the current region. Using GetCurrentValue doesn't work in this case, because there are multiple departments per region, and `GetCurrentValue("Dept.Dept")` would only return one of these values (the max department). Instead you can set a file filter template using `{Dept.Dept}`, and a filter will be created for each department value that belongs to the region.



Example multipass filter template

If departments 200, 300, and 600 all belong to the current pass region, then this file filter template will create the following filter:

```
200*.xlsx, 300*.xlsx, 600*.xlsx
```

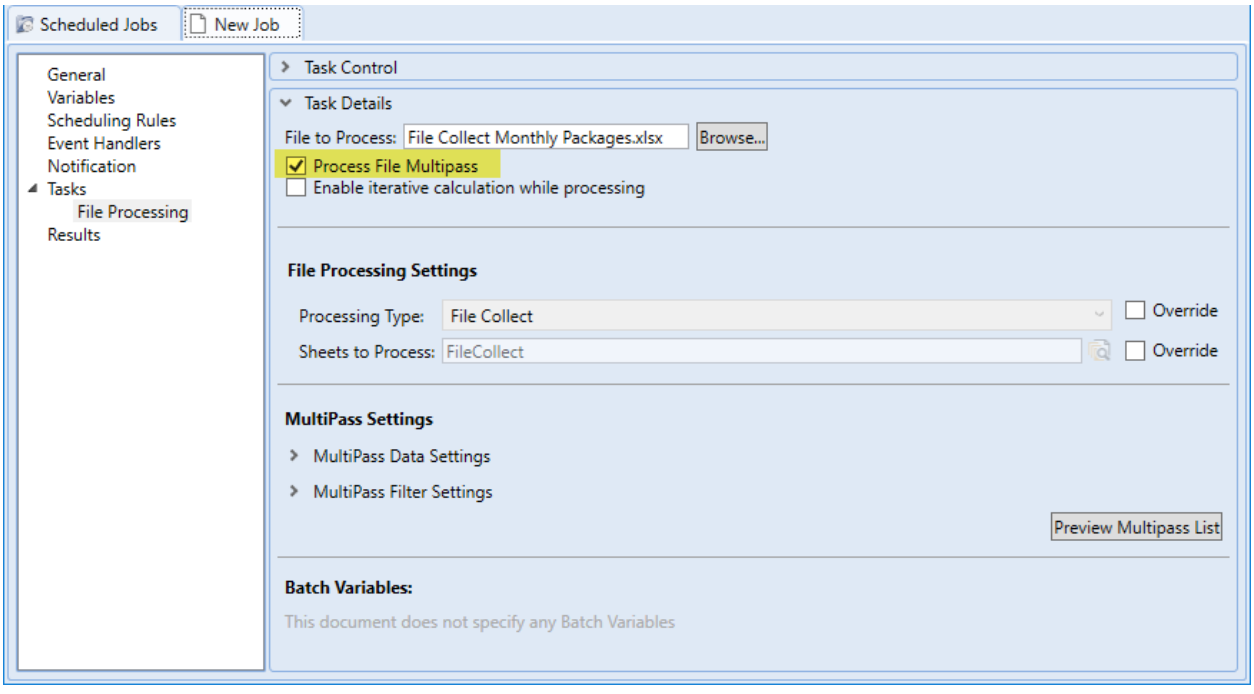
All xlsx files that start with these values will be collected into the report package for the current pass region.

Multipass processing using Scheduler or Batch

When setting up a Batch Control Sheet, the multipass options can now be used when the target file uses **File Collect** processing.

	C	D	E
5	**BATCH CONTROL SHEET**		
6			
7	**File Path**	**Enabled**	**Process Multipass**
8	\Axiom\Reports Library\Monthly Reports\Income Statement.xlsx	On	On
9	\Axiom\Reports Library\Monthly Reports\Budget Variance.xlsx	On	On
10	\Axiom\Reports Library\Monthly Reports\Forecast.xlsx	On	On
11	\Axiom\Reports Library\Monthly Reports\FileCollect\File Collect Monthly Packages.xlsx	On	On

When setting up a File Processing task in Scheduler, the **Process File Multipass** option is now available when the target file uses **File Collect** processing.



Upgrade considerations

The ability to define and use multipass processing applies to all new and existing file collect reports. However, existing File Collect Configuration Sheets do not have the new multipass filter template setting. If you want to use this setting, the best approach is to add a new File Collect Configuration Sheet to the report, using the **Add new file collect sheet** action in the File Processing task pane. Once you have configured this new sheet for multipass processing (including copying over any existing settings that remained the same from the old sheet), you can assign the new sheet as the **Sheets to Process** and then delete or archive the old sheet.

In Scheduler, File Processing tasks are treated as follows when the target file uses File Collect processing:

- When configuring new tasks, the **Process File Multipass** option is enabled by default when the target file uses File Collect processing. If you do not want to perform multipass processing, you must disable this option.
- For existing tasks, multipass processing is disabled by default, because existing files using file collect do not have multipass settings defined. These tasks will continue to perform regular processing.

Attach individual files to email using file collect

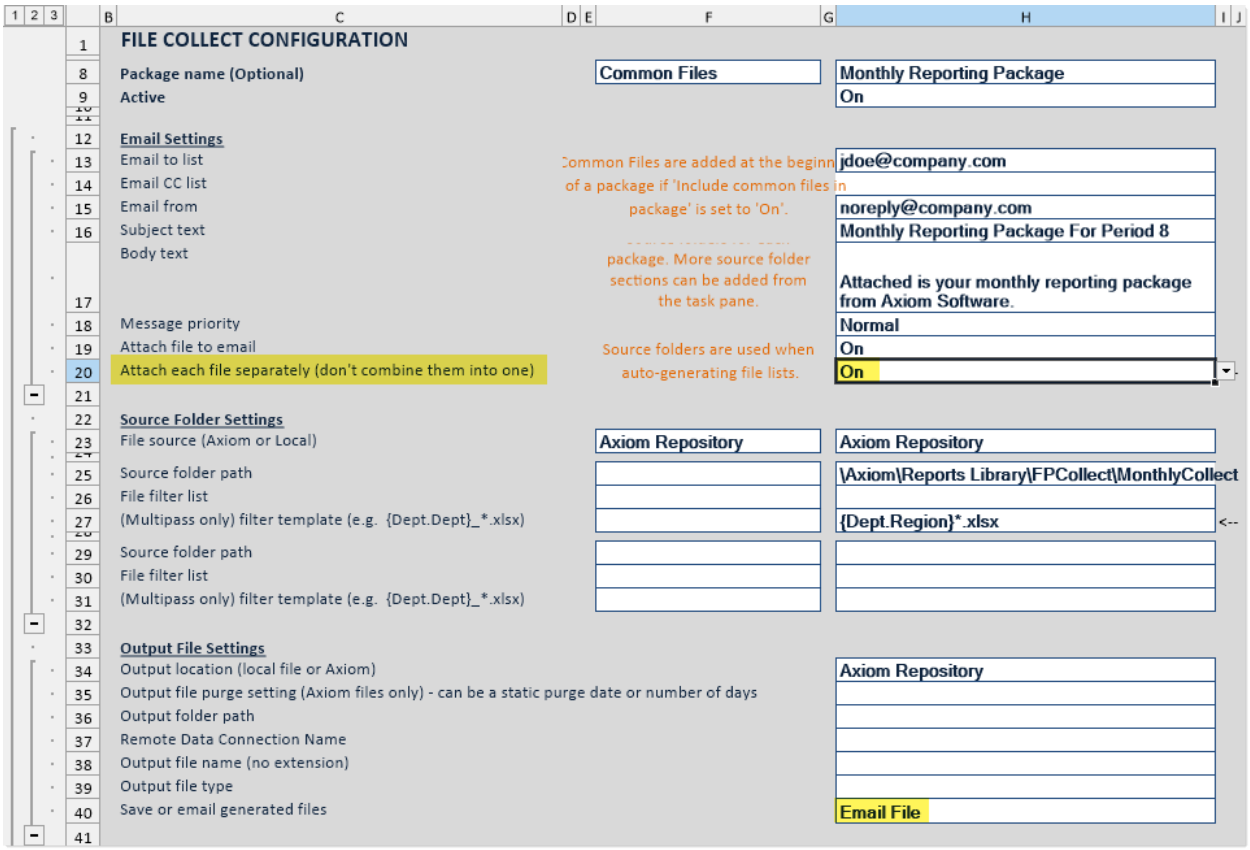
You can now configure file collect processing to attach multiple individual files to an email, instead of first collecting the file contents into a single file. The primary use case for this new feature is when the desired email recipients of the report packages do not have a one-to-one relationship with the report packages.

For example, imagine that you need to generate one report package per region and then deliver those packages to the appropriate VPs. If each region has a unique VP, then this can easily be done in one file collect operation. But if some VPs are responsible for multiple regions, then normal file collect processing will result in the VPs getting multiple emails (one for each region). If instead you want those VPs to get one email with multiple region attachments, then you can perform file collect processing in two phases:

- Phase one to collect and create the report packages for each region, and save them to a file location.
- Phase two to individually attach the relevant region packages to an email for each VP.

This new behavior would typically be used in conjunction with the new ability to perform multipass processing for file collect, but can also be used with regular file collect processing as needed.

To enable this behavior, use the new option **Attach each file separately** in the **Email Settings** section. If this option is enabled, and if **Save or email generated files** is set to **Email File**, then the files in the file list are attached to the email as individual files instead of being collected into a single file.



New option to attach files separately when emailing

For a detailed example of how this feature might be used—from creating the initial snapshots, to collecting the report packages, to attaching the files to an email—see the following topic in Axiom Software Help: *File collect examples* (AX1773).

File groups

This section details the new features and enhancements made to file groups.

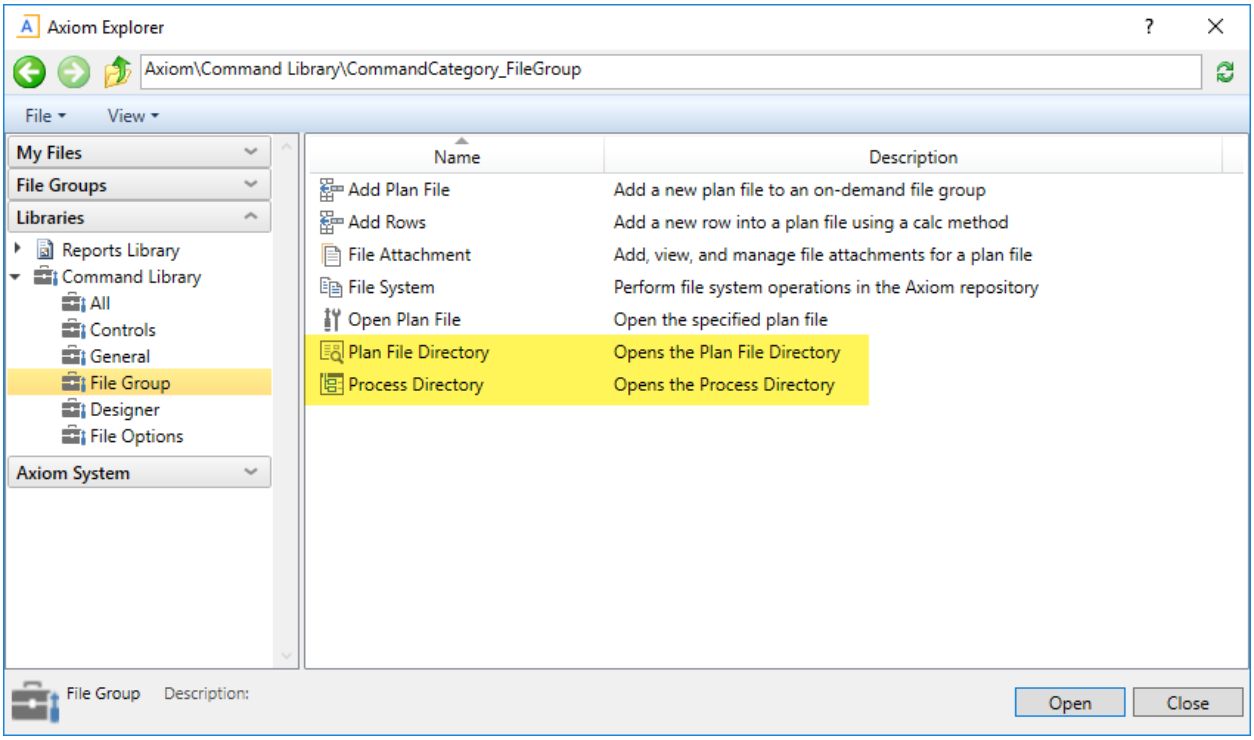
New commands to open file group directories

You can now open the Plan File Directory web page and the Process Directory web page using commands, instead of needing to manually create the URL. This makes it much easier to use these web pages as part of any planning solution.

- The Plan File Directory is a built-in web page that displays the plan files a user has rights to access in a file group. Users can open the plan files from this page.
- The Process Directory is a built-in web page that displays the process status of all plan files the user has rights to access in a file group. Step owners can complete process tasks from this page and review process details.

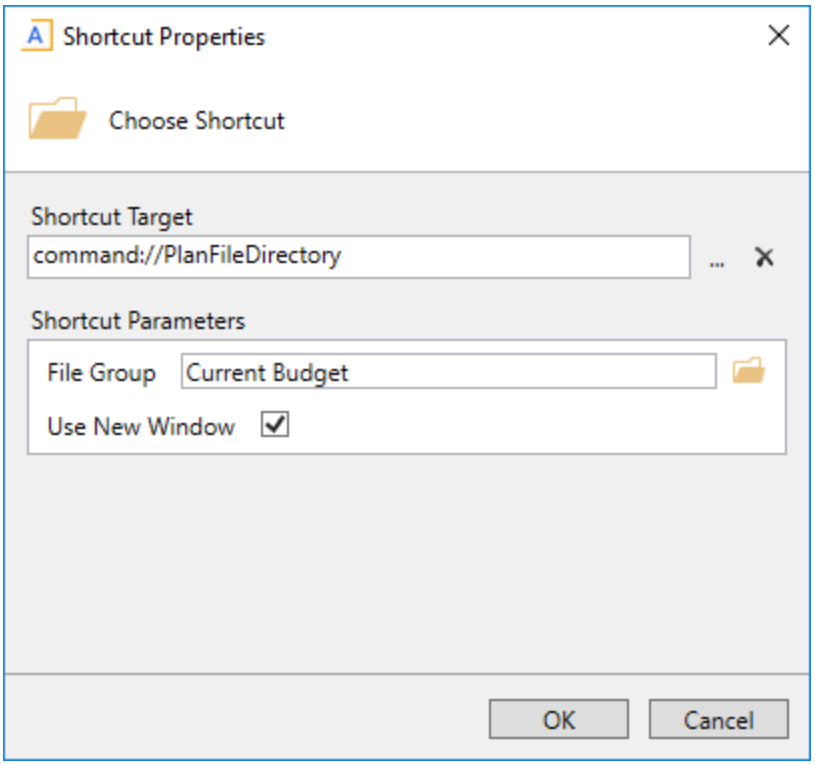
The commands can be used in any of the following features, to provide users with easy access to these pages:

- Axiom forms
- Ribbon tabs
- Task panes (for use in the Desktop Client or as web navigation)



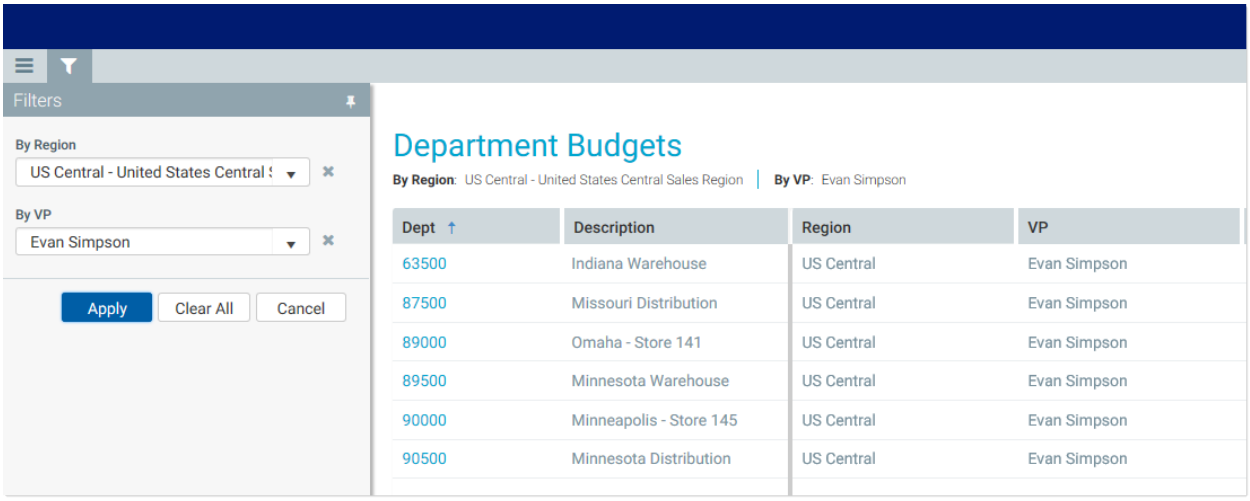
New commands in the Command Library

To configure the command, you specify the target file group and whether the directory should open in a new window. File group alias names can be used to dynamically update the target file group.



Example command parameters

When the command is used, it automatically opens the directory for the specified file group.



Example web directory page

New way to return values from the plan code table in plan files

Plan files often need to reference multiple related values from the plan code table—for example, the plan code description and other attributes such as department type, region, or VP. You can now return these values using a lightweight method instead of needing to use data queries such as GetData.

When a plan file is opened, all associated values from the plan code table are now automatically loaded into document memory and can be retrieved using the new GetPlanItemValue feature. Using GetPlanItemValue instead of GetData eliminates the need to make round-trip server calls to the database to display this information in plan files. If you have large plan files with many data queries, using this feature along with other optimization features (such as time-stamped Axiom queries or batched queries) can help improve file performance.

GetPlanItemValue is available as an Axiom function and as a data lookup. It uses the following syntax:

`GetPlanItemValue("`*`ColumnName`*`")`

Where *ColumnName* is the name of the column in the plan code table for which you want to return the value. For example, `GetPlanItemValue("Description")` returns the value in the Description column for the current plan code.

The following screenshot shows an example of GetPlanItemValue as a data lookup. The **Insert Data Lookup Data Source** wizard and the **Data Source Assistant** have been updated to include this new row type.

	C	D	E	F	G
5					
6		[DataLookup;Values]	[result]	[iserror]	[columnname]
7		[GetPlanItemValue]	25000	FALSE	Dept
8		[GetPlanItemValue]	Finance	FALSE	Description
9		[GetPlanItemValue]	Master Budget Template	FALSE	Template
10		[GetPlanItemValue]	Corporate	FALSE	WorldRegion

Example GetPlanItemValue data lookups

Keep in mind that once the associated values are loaded into document memory, they are not subsequently updated during the current document session. If the values in the plan code table change during the session, GetPlanItemValue will not reflect that change until the file is closed and reopened. If you need a particular value to update during the document session, you should use GetData to retrieve that value instead.

Additional file group enhancements

The following additional enhancements were made to file groups.

File locking for virtual spreadsheet plan files

The file locking behavior and read-only designation has been changed for virtual spreadsheet plan files. This change only applies if:

- The option **Use Virtual Plan Files** is enabled in the file group properties.
- The plan files are opened as spreadsheets in the Desktop Client (not used as Axiom forms).

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 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.

When a user opens a virtual spreadsheet plan file, the file is now flagged as read-only 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.

Additionally, if a user has Read/Write access to the virtual spreadsheet plan file, Axiom Software now locks the plan file when the user opens it with the ability to save data, in order to prevent other users from saving data in the file at the same time. When process management elevates user permissions, it grants Read/Write access with Allow Save Data, so the new behavior will automatically apply to step owners. 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 to the virtual plan files instead of just Read-Only with Allow Save Data.

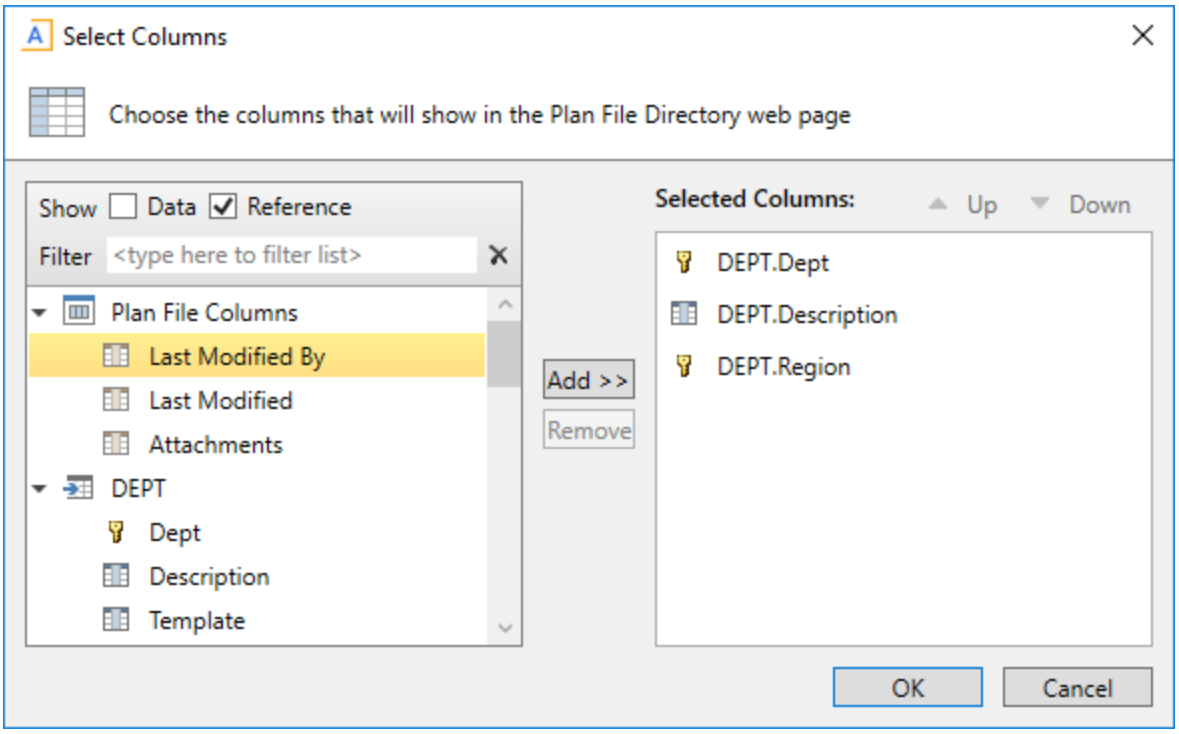
NOTE: This behavior does not apply to virtual form-enabled plan files. File locking does not apply to Axiom forms, and Axiom forms are not flagged as read-only. If you want to control data saves in form-enabled plan files, the existing "save locking" feature for Axiom forms can be used.

System columns available in the Plan File Directory

When configuring display columns for the Plan File Directory of a file group, you can now optionally add the following system columns:

- **Last Modified By**
- **Last Modified**

These columns are listed in the **Plan File Columns** node.



New display columns available for the Plan File Directory

Axiom files

This section details the new features and enhancements that impact general Axiom file design.

New function to return document information

A new Axiom function, GetFileSystemInfo, is available to return information about a specified document. This function should be used as the preferred alternative to building an Axiom query to the Axiom.FileSystemInfo table, because queries to that table can negatively impact system performance.

The GetFileSystemInfo function is similar to the existing GetDocumentInfo function. The GetDocumentInfo function returns information about the current document, whereas the GetFileSystemInfo function returns information about any document in the Axiom file system.

The GetFileSystemInfo function uses the following syntax:

```
GetFileSystemInfo("Code","Path")
```

Where *Code* specifies the type of information to return about the document, and *Path* is the full path to the document in the Axiom file system.

The available codes to return information are as follows:

Code	Description
Description	Returns the description of the file, if defined (blank otherwise).
DocumentID	Returns the database ID of the file.
DocumentType	Returns the document type of the file (such as Report or Template).
FolderID	Returns the ID of the folder where the file is located.
LockedByUser	Returns the login name of the user who currently has the file lock, if applicable (blank otherwise).
RecordCreatedBy	Returns the login name of the user who created the file.
RecordCreatedDTM	Returns the date and time the file was created.
RecordModifiedBy	Returns the login name of the user who last modified the file.
RecordModifiedDTM	Returns the date and time the file was last modified.

For example:

```
=GetFileSystemInfo("DocumentID", "\Axiom\Task Panes
Library\Navigation.axl")
```

This example returns the database ID of the designated file. For example: 219.

```
=GetFileSystemInfo("DocumentID", GetFileGroupProperty("DriversPath",
"Current Budget")&"\GeneralDrivers.xlsx")
```

This example uses GetFileGroupProperty to return the path to the Drivers folder for the specified file group, and then returns the document ID of the designated driver file. In this case Current Budget is a file group alias name, so the function updates dynamically as the target of the alias changes.

```
=GetFileSystemInfo("RecordModifiedDTM", "\Axiom\Reports
Library\Forms\Dashboard.xlsx")
```

This example returns the last modified date/time of the specified file.

New function to apply views on demand

Using the new function ReapplyCurrentViews, users can reapply the current view on demand by double-clicking a cell. The purpose of this function is to provide the ability to "expand" and "collapse" groupings in spreadsheet Axiom files by using view tags—for example, to dynamically show and hide rows.

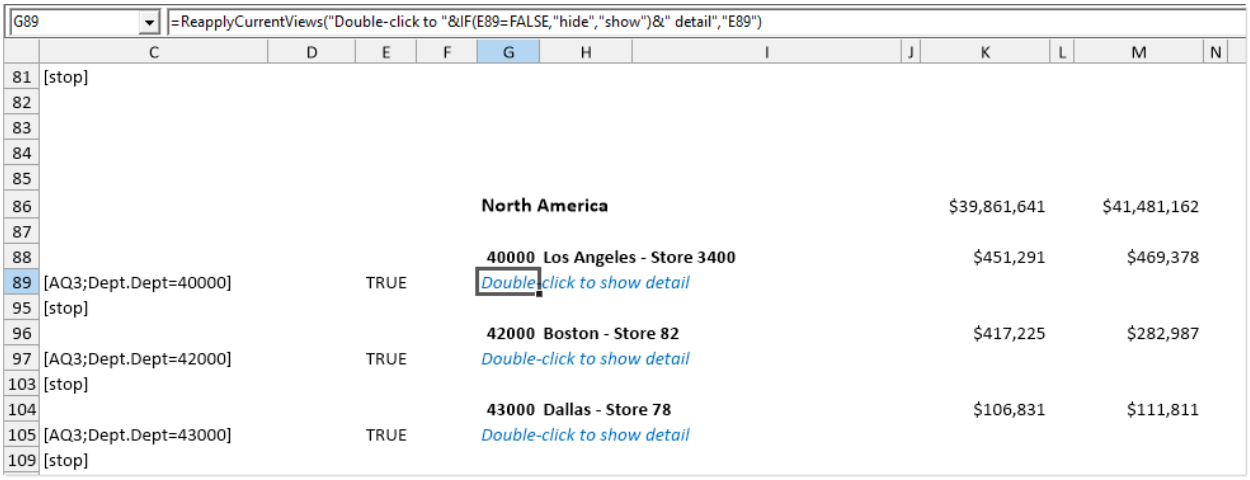
When a user double-clicks the cell with the function, a Boolean value (True/False) is toggled in a target cell, and then the current views are reapplied. The view tags can use a formula that references the target cell, to either show or hide the rows (or columns) based on the current cell value.

The function ReapplyCurrentViews uses the following syntax:

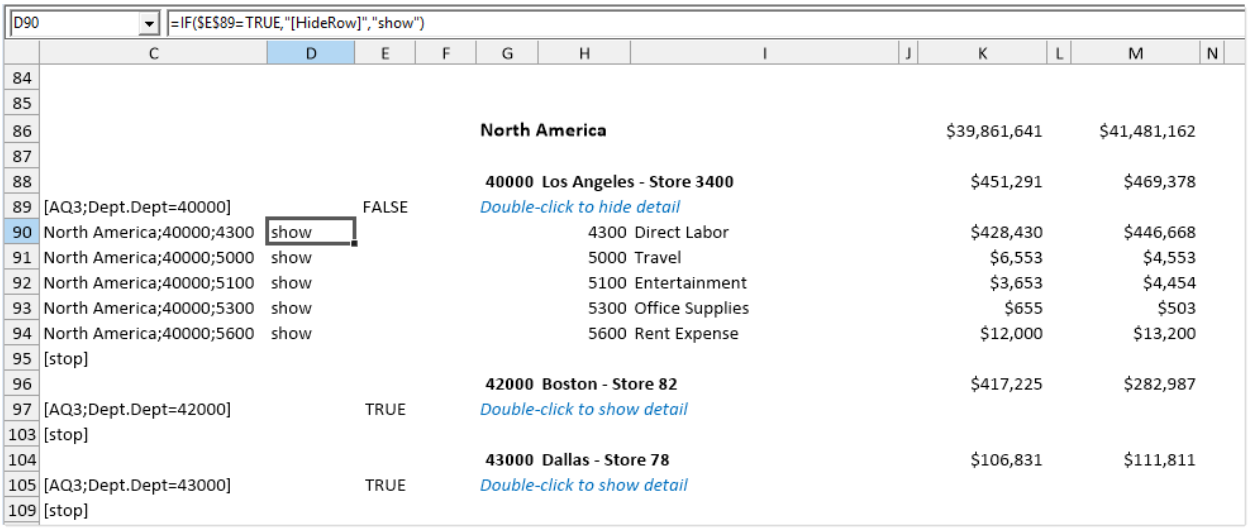
```
ReapplyCurrentViews("DisplayText","TargetCell")
```

Where *DisplayText* is the text to display in the cell, and *TargetCell* is the cell where the Boolean value is toggled.

The following example screenshots illustrate how this function can be used. In this example, the file has a defined **Initial Dynamic View** that is applied on open, and this view hides the detail rows. Each section has been set up with a ReapplyCurrentViews function.



When a user double-clicks the cell with the function, the value of the target cell is toggled—in this case, cell E89 goes from True to False. The file is calculated so that any formulas referencing this cell are now updated, and then views are reapplied. In this example, the rows in this section are now shown because the formula is resolved to no longer result in `[HideRow]` tags.



	C	D	E	F	G	H	I	J	K	L	M	N
84												
85												
86					**North America**				$39,861,641		$41,481,162	
87												
88					**40000**	**Los Angeles - Store 3400**			$451,291		$469,378	
89	[AQ3;Dept.Dept=40000]		FALSE		*Double-click to hide detail*							
90	North America;40000;4300	show				4300	Direct Labor		$428,430		$446,668	
91	North America;40000;5000	show				5000	Travel		$6,553		$4,553	
92	North America;40000;5100	show				5100	Entertainment		$3,653		$4,454	
93	North America;40000;5300	show				5300	Office Supplies		$655		$503	
94	North America;40000;5600	show				5600	Rent Expense		$12,000		$13,200	
95	[stop]											
96					**42000**	**Boston - Store 82**			$417,225		$282,987	
97	[AQ3;Dept.Dept=42000]		TRUE		*Double-click to show detail*							
103	[stop]											
104					**43000**	**Dallas - Store 78**			$106,831		$111,811	
105	[AQ3;Dept.Dept=43000]		TRUE		*Double-click to show detail*							
109	[stop]											

If the user double-clicks the cell again, the target cell is toggled back to True and the formulas resolve to display `[HideRow]` tags—resulting in the rows being hidden when the views are reapplied.

Run save-to-database processes concurrently

Save-to-database processes within an Axiom file now run concurrently by default, instead of sequentially. This may improve file performance in files with many save-to-database processes, or when a handful of save-to-database processes save many records.

Save-to-database blocks are still identified and triggered using the same order as in previous versions:

- Each sheet is processed in the order they are listed on the Control Sheet.
- Within a sheet, Axiom searches for save-to-database tags by row from left to right, starting at the first row and then moving down. If the tags are numbered, then the save blocks are processed in numeric order, otherwise they are processed in the order found.

When Axiom Software identifies a save block to be processed, it now triggers a background task to handle the save instead of waiting for the save to finish. It then immediately moves on to identify and trigger the next save block, so that multiple saves are run concurrently.

In order to automatically handle dependent save-to-database processes, the following exceptions apply:

- **Reference tables**: If the destination table for the save block is a reference table, Axiom Software triggers the background task for the save but then waits for all current tasks to complete before moving on to the next save block. This behavior is intended to handle cases where records are saved to a reference table, and then a subsequent save references the new or updated records.

- **Save Type 4**: If the save block is SaveStructure2DB (Save Type 4) instead of Save2DB (Save Type 1), Axiom Software triggers the background task for the save but then waits for all current tasks to complete before moving on to the next save block. This behavior is intended to handle cases where subsequent save blocks depend on the changes made using Save Type 4.
- **Axiom query**: If the save block triggers an Axiom query to be run before the save-to-database occurs, Axiom Software waits for all current tasks to complete before continuing. Once all tasks are complete, Axiom Software runs the specified Axiom query and then triggers the background task for the save. This behavior is intended to handle the case where the Axiom query is returning records that may be impacted by a previous save block.
- **Same data table**: If the destination table for a save block is the same destination data table as a previous save block, Axiom Software triggers the background task for the save but does not start processing it until the task for the previous save block to the table has been completed. This behavior is intended to prevent database errors that could result if both save blocks attempted to add or update the same records at the same time.

Additionally, because save blocks are now processed concurrently, the point at which the save-to-database process stops due to error may be different. In previous releases, Axiom Software stopped processing save blocks after encountering the first error. Because multiple save blocks may already be in process when an error is encountered, those processes will continue. However, no new save blocks will begin processing after the error.

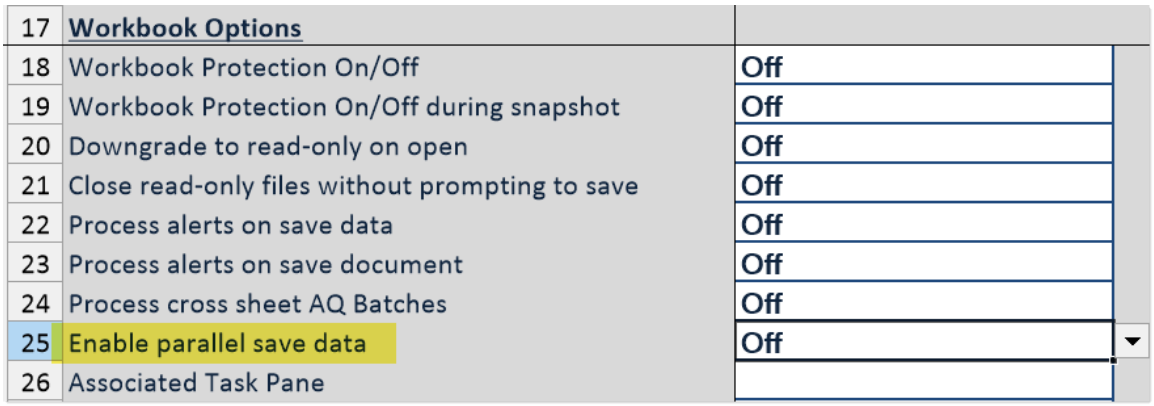
NOTE: The parallel save behavior only applies when a user executes a save from within a file, to improve performance when communicating between the client and the application server. When a file is processed server-side using Scheduler and a save-to-database occurs, all saves are executed sequentially.

Upgrade considerations

When you upgrade, the new parallel save behavior is automatically enabled and applies to all Axiom files. This new behavior has been carefully designed so that there should be no adverse impacts to your existing save-to-database processes. In some cases, you may see a performance improvement.

However, because of the wide variety of use cases and options for save-to-database processes, it is possible that there may be an edge case that the new behavior does not handle. If an edge case is identified, it is possible to disable the behavior on a per file basis, or for the entire system:

- **Per File**: To disable the new behavior for a particular file, set **Enable parallel save data** to **Off** on the Control Sheet. For existing files, you must upgrade your Control Sheet to gain access to this setting.



New setting to disable the default parallel save behavior for a file

- **System-Wide**: To disable the new behavior for an entire system, set the system configuration setting **ParallelSaveEnabled** to FALSE. For information on how to modify the system configuration settings, see the following topic in Axiom Software Help: *System configuration settings* (AX2015).

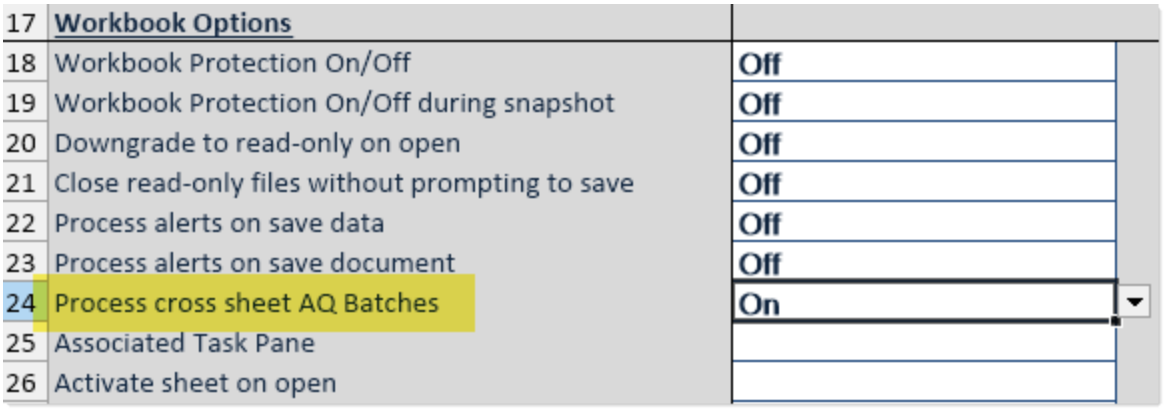
If you do identify an edge case, please inform us about it so that we can consider further changes to the parallel save behavior.

Run Axiom query batches across sheets

When using batched Axiom queries, it is now possible to configure batches that include queries from one or more sheets, to execute those queries concurrently instead of sequentially. Cross-sheet batching may be useful when the file contains queries on several different sheets, many of which can be run at the same time.

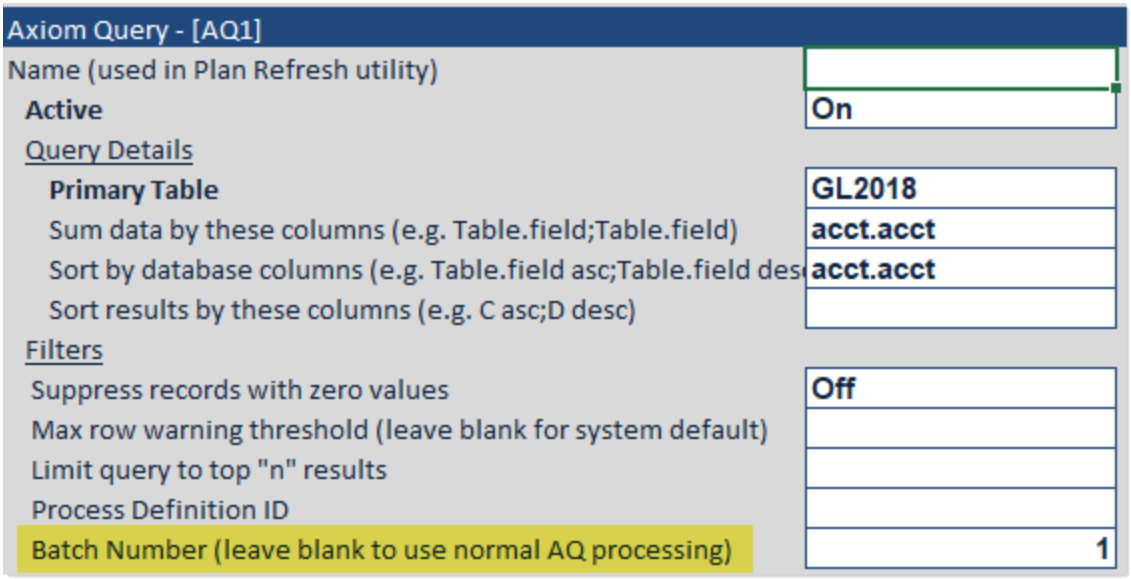
By default, Axiom query batches are defined and run per sheet. Each sheet is still processed in order, with the batched queries on the sheet executed first in parallel, followed by the non-batched sequential queries.

If desired, you can enable cross-sheet batching so that batches can include Axiom queries on any sheet. To do this, set **Process cross sheet AQ Batches** to **On**. This setting is located in the **Workbook Options** section of the Control Sheet.



New setting to enable cross-sheet batching

When this setting is enabled, the batch numbers assigned to Axiom queries are evaluated against all sheets. For example, if a query is assigned to batch 1 on Sheet1 and on Sheet2, those queries are run concurrently within batch 1, even though they are defined on different sheets. Batch numbers are assigned per query in the Axiom query properties, in the **Query Details** section.



Existing setting to specify batch number

Cross-sheet batching works as follows:

- Queries in batch 1 are processed first. This occurs regardless of which sheets the queries are on. Queries are processed in parallel, which means:
    - The database queries for all Axiom queries in the batch are made at the same time.
    - Axiom query data is placed in the sheet in the order that the database queries are completed. For example, if AQ1 and AQ2 are in the same batch, but the database query for AQ2 completes first, then data for AQ2 is placed in the sheet first.
- Queries in batch 2 are processed next, and so on, until all batches are completed.
- After all batches are completed, any remaining Axiom queries are processed using normal per sheet processing, in ascending order based on query number.

Cross-Sheet Example

Imagine that the Axiom queries in a file are configured as follows:

Sheet1		Sheet2	
Query	**Batch**	**Query**	**Batch**
AQ1	1	AQ1	2
AQ2	2	AQ2	1
AQ3	1	AQ3	2
AQ4	2	AQ4	
AQ5			

When this file is processed, the queries are run as follows:

- The queries in batch 1 are processed first, in parallel. This includes AQ1 and AQ3 from Sheet1, and AQ2 from Sheet2.
- The queries in batch 2 are processed next, in parallel. This includes AQ2 and AQ4 from Sheet1, and AQ1and AQ3 from Sheet2.
- Normal per-sheet query processing now begins. AQ5 on Sheet1 is processed first, then AQ4 on Sheet2.

Cross-sheet batching otherwise has the same limitations as regular batching. For more details on how Axiom query batching works, see the following topic in Axiom Software Help: *Batch processing for Axiom queries* (AX1743).

Upgrade considerations

By default, the cross-sheet batching option is disabled. This means that all existing files with Axiom query batch numbers will continue to execute sheet-by-sheet, as they did in previous versions.

If you want to enable cross-sheet batching for an existing file, upgrade the Control Sheet and then enable the option. If you do this, you should review and update batch numbers as needed.

Additional Axiom file enhancements

The following additional enhancements were made to Axiom files.

New data lookup row types

In addition to the new GetPlanItemValue, the following new row types have been added to data lookups:

- `GetTableInfo`: Returns information about a specified table. This supports the same codes as the existing GetTableInfo function. Additionally, both the function and the data lookup now support a

new code of `Exists`, which can be used to determine whether a particular table name exists in a particular system (True/False).

- `GetFeatureInfo`: Returns information about an installed product feature. This supports the same codes as the existing GetFeatureInfo function.

The **Insert Data Lookup Data Source** wizard and the **Data Source Assistant** have been updated to include these new row types.

Option to suppress collapse behavior for RunAxiomQueryBlock function

When using the function RunAxiomQueryBlock, you can now optionally configure the function to *not* zero or "collapse" populated blocks. This behavior is intended to support use cases where the Axiom query rows contain user inputs, and you do not want to accidentally delete these inputs before saving.

The function now has a fourth parameter of SuppressCollapse, which by default is False but can be set to True as needed. The function parameters are now as follows:

```
RunAxiomQueryBlock("DisplayText", "AxiomQueryName", "NestedQueryNames",
SuppressCollapse)
```

For example:

```
=RunAxiomQueryBlock("Double-click to see detail", "Populate Detail", ,
True)
```

This example could be used to populate a set of detail rows into a plan file, for the user to input data. When the user double-clicks the function for the first time, the Axiom query is run and the detail rows are populated. If the user double-clicks the function again when the Axiom query block is already populated, no action occurs.

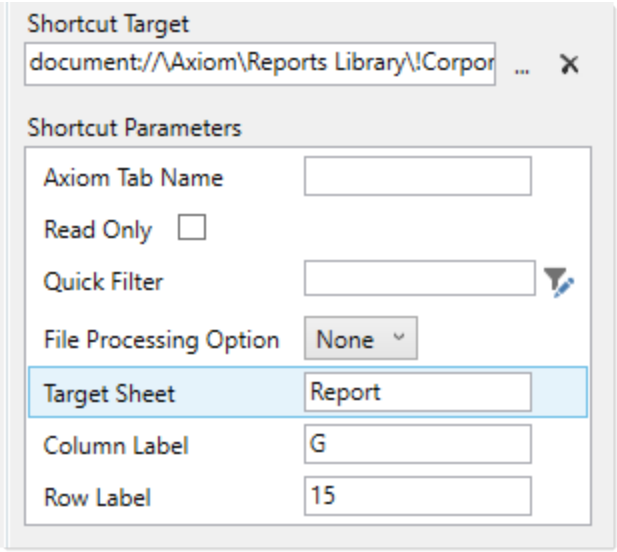
Open an Axiom file to a specific location from a task pane or ribbon tab

When linking to an Axiom file in a custom task pane or ribbon tab, you can now specify a sheet, column, and row location to make active when the file is opened. The following optional shortcut parameters are now available when the target file is a spreadsheet Axiom file:

Item	Description
Target Sheet	Specifies a sheet in the workbook to make active when the file is opened from the task pane or ribbon tab. This overrides the **Activate sheet on open** setting defined in the file.

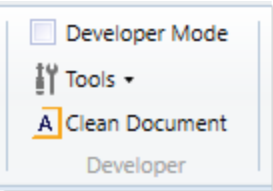
Item	Description
Column Label Row Label	Specifies a column and row location in the target sheet to make active when the file is opened from the task pane or ribbon tab. There are two options to define the column and row location: • **GoTo bookmarks**: If the target sheet contains GoTo bookmarks, you can enter the GoToColumn name as the **Column Label**, and a GoTo tag name as the **Row Label**. When the file is opened, the cursor is placed in the same location as if the user had selected that GoTo label from the GoTo menu. • **Cell reference**: If you want the cursor to start at cell F22, you can enter F as the Column Label and 22 as the Row Label. If no target sheet is specified, the column label and row label are ignored.

In the following example, the target file is opened to the Report sheet, with the cursor at cell G15.



Other enhancements

The Axiom Designer ribbon tab has a new tool named **Clean Document**, to clear the **Last refresh time** for time-stamped Axiom queries and any data lookup results. This tool allows file developers to reset and run queries as needed.



Imports

This section details the new features and enhancements for import utilities.

Import into tables with an identity column

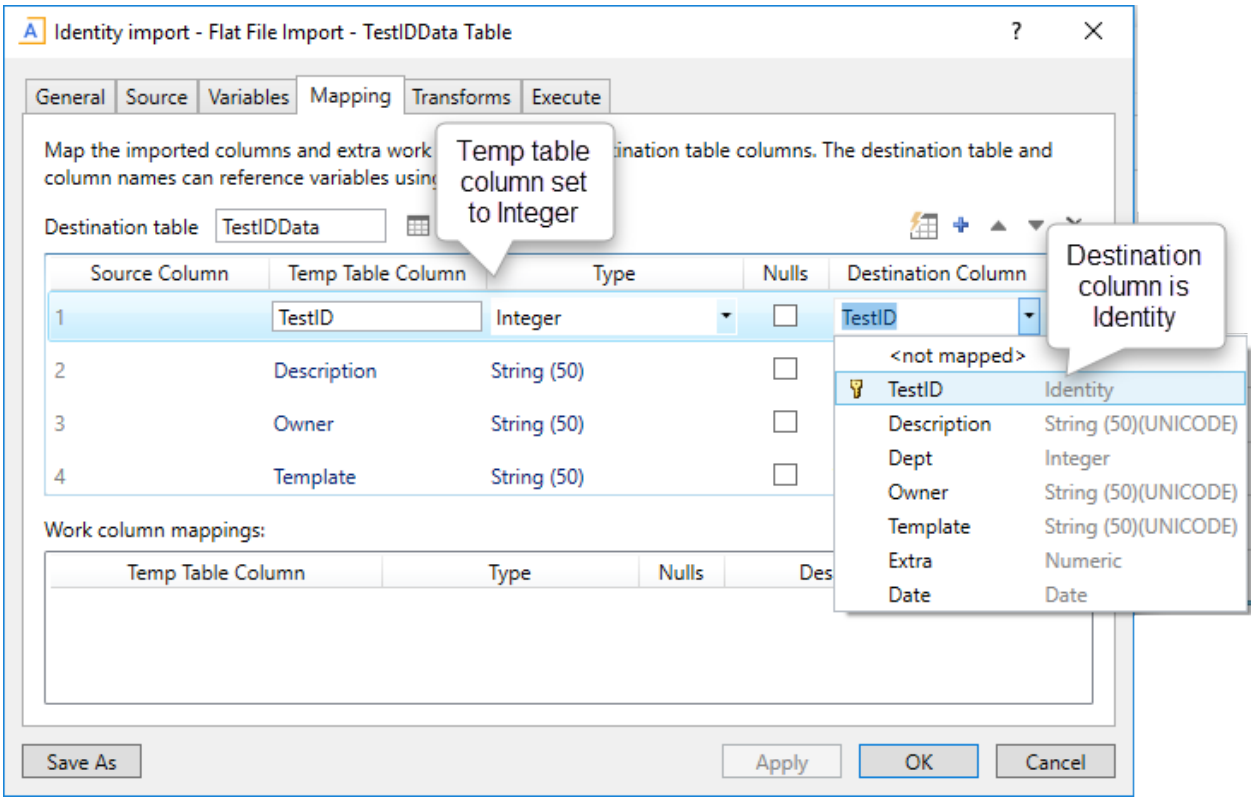
Imports now support the ability to import data into tables with an identity column. In previous versions, this was not possible using the standard Import Wizard interface. Columns that use the identity data type contain automatically-generated, unique ID numbers.

The identity column can be mapped as part of the import, or left unmapped. The decision of how to handle this depends on whether you are updating existing data or only creating new records, and whether you want new records to use automatically-generated numbers or specific numbers.

Updating existing data with an identity key column

If the import is updating existing data and the table contains an identity key column, the identity column must be mapped. In order to update existing data, the import must be able to match the value in the import temp table with an identity key in the destination table.

When mapping a temp table column to an **Identity** column in the destination table, the data type of the temp table column must be set to **Integer**. (If the identity column is **Identity32**, the temp table column should be set to **Integer32**.)



Mapped identity column in an import

Creating new records with an identity key column

If the import is creating new records in the destination table, you can opt to use automatically-generated ID numbers in the identity column, or you can specify the ID numbers.

If you want to use automatically-generated ID numbers, there are two ways to accomplish this:

- **Leave the identity column unmapped.** When the identity column is unmapped, all records in the import data are created as new records with automatically-generated ID values. This configuration is appropriate when the only purpose of the import is to create new records. If the import data contains a mix of new and updated records, then the identity column cannot be left unmapped.

  This is also the only use case where it is possible to leave a key column unmapped in an import.

- **Map the identity column, but leave the column blank for new records.** When the identity column is mapped, but the temp table column is left blank, new records are created for the blank values using automatically-generated ID values. This configuration is appropriate when the import data contains a mix of new and updated records. If the column contains an existing identity value, the existing record is updated.

In some cases, you may have a need to create new records using specific identity values. You can do this by mapping the identity column, and populating the import data with the desired identity values. When new records are created in the table, they will use the specified values instead of using automatically-generated values. When using this approach, keep in mind the following:

- In order to create new records, the values in the temp table column must not already exist in the destination table. If a temp table value matches an existing value, then the existing record is updated instead of creating a new record.
- The "seed" value for the identity column will be reset to the largest inserted value, instead of continuing where it left off. For example, if the last auto-generated value was 20, but you import a specific new value of 80, the next auto-generated value will start at 81.
- It is not possible to create new records with a mix of automatically-generated ID values and specific values. If you want to create any new records with specific values, then all new records in the import data must be assigned specific values. The temp table column cannot be left blank when using this configuration.

Related changes

It is now also possible to create new records using specific identity values when saving to the table using Save Type 1. If the identity column contains specific values that are not already present in the table, new records will be created using those values. This approach has the same limitations as noted previously for imports—the identity "seed" value will be reset to the largest inserted value, and the save set cannot also contain blank values. You must choose whether to create new records by using blank values (resulting in automatically-generated ID values) or by using specific values.

Additionally, the process for creating new identity records when using Open Table in Spreadsheet has changed. In previous releases, you had to populate the identity column with a numeric value that was not already used in the table. When the save occurred, this value was ignored and instead an automatically-generated value was applied. Going forward, you must leave the identity column blank in order to create a new record with an automatically-generated value. Note that it is not possible to create new records with specific values when using Open Table in Spreadsheet.

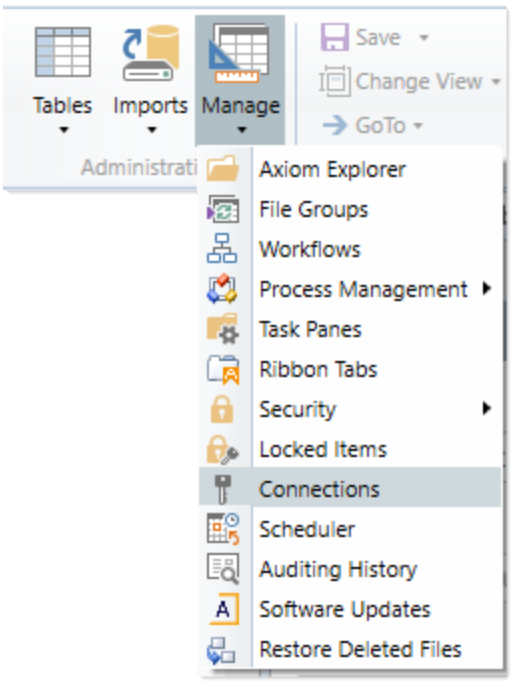
Additional import enhancements

The following additional enhancements were made to imports.

Manage Ellucian Ethos import connections

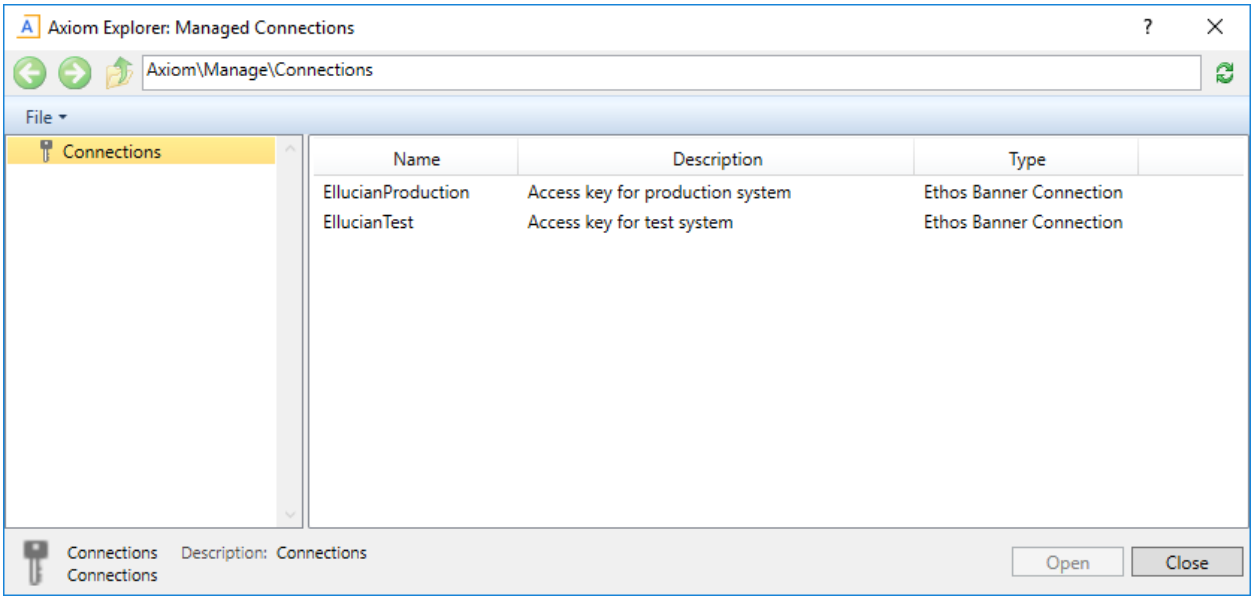
In 2019.2, Axiom Software introduced the ability to import data from Ellucian Banner systems. In order to configure imports to connect to your Ellucian system, you create a named connection which stores the necessary Ellucian Ethos access key.

These connections can now be managed within the separate **Managed Connections** dialog. To access this dialog, select **Manage > Connections** from the default **Axiom** ribbon tab.



New menu option to manage connections

Using this dialog, you can create, edit, and delete connections without needing to open individual imports. Any connections created here can be used in Ellucian imports. This dialog is only available if the ability to import data from Ellucian systems has been enabled for your system.



Example Managed Connections dialog

For convenience, you can still create new Ellucian Ethos connections from within the Import Wizard as needed.

Visibility of aggregation option

The option **Aggregate rows on final save** is now hidden on the **General** tab of the **Import Wizard** if it does not apply to the import configuration, such as when the destination table is a reference table. In previous releases the setting was always present, but ignored if it was inapplicable.

Additional enhancements

Null values allowed in non-key validated columns

It is now allowed to store null values in a validated column, meaning that the column can contain valid values from the assigned lookup column and null (blank) values. If you want to allow null values in a validated column, the default value of the column must be null. Generally speaking, this should only be done when the null values have a specific understood meaning for your data, such as unassigned or inapplicable.

Although Axiom Software used to require the default value of a validated column to be a valid lookup value when you first assigned the lookup column, it was possible to later edit the validated column to change the default value to null. So this behavior change could potentially affect existing columns, though the situation should be rare. In previous versions there would have been no reason to change the default value to null, because null would not have been allowed when saving and would have caused an error.

If you have an existing validated column with a null default value, and the column is omitted from the save when creating new records, or if it is included in the save but left blank (for non-string columns), the save now uses the default value of null instead of causing an error due to an invalid lookup value.

NOTE: As in previous versions, we do not recommend using a null default value with a string column, whether it is validated or not. The behavior of string validated columns when using a null default value is inconsistent (due to the inability to differentiate between null and empty string in a spreadsheet), and should be avoided.

Miscellaneous

- When using Save Type 4 with dependent saves—such as to create a column sequence and then create column sequence items—it is no longer necessary to execute this save using two separate passes. As long as the SaveStructure2DB tags are in the correct order within the file—so that the column sequence is created before attempting to create the items—the entire sequence can be created in one save-to-database process.

- The Run QA Diagnostics test named **Analyze Axiom UDFs** now displays a warning if it finds any GetData functions in a form-enabled document. GetData functions should be avoided in forms whenever possible, because they will fire repeatedly during each form update.
- The result summary page for Run QA Diagnostics tests has been updated to show total errors and warnings. The error totals are color-coded to reflect critical, major, and minor errors.