***TABLE OF CONTENTS***

[1 Introduction 3](#_Toc14171778)

[1.1 Overview and Purpose 3](#_Toc14171779)

[1.2 Definitions and Acronyms 3](#_Toc14171780)

[2 Environment 3](#_Toc14171781)

[2.1 User Folder Structure 3](#_Toc14171782)

[2.2 Alteryx User Environment Guidelines 3](#_Toc14171783)

[3 Workflow Development Best Practices 4](#_Toc14171784)

[3.1 Guiding Principles 4](#_Toc14171785)

[3.2 New Workflow Definition 5](#_Toc14171786)

[3.3 Tool-Level Best Practices 6](#_Toc14171787)

[3.4 Tool Alignment 7](#_Toc14171788)

[4 Workflow Documentation 7](#_Toc14171789)

[4.1 Internal 7](#_Toc14171790)

[4.2 Container Usage 8](#_Toc14171791)

[4.3 External 8](#_Toc14171792)

[5 Standard Macros 9](#_Toc14171793)

[6 Unit Testing 9](#_Toc14171794)

[7 Workflow Version Control 10](#_Toc14171795)

[8 Server / Gallery 10](#_Toc14171796)

[8.1 Location 10](#_Toc14171797)

[8.2 Permissions 10](#_Toc14171798)

[9 Integration with External Tools 10](#_Toc14171799)

[9.1 Tableau 10](#_Toc14171800)

[9.2 R 10](#_Toc14171801)

[9.3 Python 10](#_Toc14171802)

[10 Contributors 10](#_Toc14171803)

[11 Version History 12](#_Toc14171804)

# Introduction

## Overview and Purpose

To support the widespread deployment of Alteryx through the organization, the following document defines common patterns and guidelines to encourage consistency of output from all functions and developers.

## Definitions and Acronyms

| **Term** | **Definition** |
| --- | --- |
| **Workflow** | Alteryx structure designed to take input data, perform specific steps of data preparation, summarization and analysis to produce a data output or analytic insight. |
|  |  |

# Environment

## User Folder Structure

The following user folder structure is recommended to facilitate sharing of Alteryx components.

<Home Directory>\

 \Workflows

 \Macros

 \Scripts

 \Documentation
 \Logs

 \Common

 \Workflows

 \Macros

 \Scripts

 \Documentation

Individual users work products should be stored in the top structures. Common components should be sourced from the central repository for the organization.

## Alteryx User Environment Guidelines

When opening Alteryx for the first time, there are a number of settings which can be chosen if desired. The following user settings can tailored. This is generally an individual choice but some recommended adjustments to the settings to consider are below.

* + Turn the Snap to Grid off: Unchecking really helps to make straight lines in the modules and keep track of where things go.
		- Options->User Settings->Edit User Settings->Document
		- Uncheck 'Snap to Grid'
	+ Turn the XML view on. With this checked, you have another option to view the tool's XML in the properties window. Very helpful when trying to determine previous settings, Allocate settings, etc...
		- Options->User Settings->Edit User Settings->Advanced
		- Check 'Display XML in Properties Window'
	+ Turn Logging on and be aware of the directory location. The messages that appear in the message window are informative but when working on more complex workflows, there will be additional information in the logs like tool performance.
		- Options->User Settings
		- Set “Logging Directory” to “Logs” in your user directory (create the Folder if not available)

# Workflow Development Best Practices

## Guiding Principles

The flexibility of the tool invites a variety of solutions to the same problem. As such, the following guiding principles are suggested for development of workflows. In general, without trying to pre-optimize, try to keep it lean for performance. General illustrations of this concept follow.

* **Don't handle data you're not going to process.** Place a SELECT tool immediately after your INPUT limiting your data stream to the fields that you are concerned with. Add descriptions. Even if you are not currently limiting your data stream, adding this tool now could save you headaches down the road if the architecture of the INPUT is outside your control.
* **Use SQL browser to limit to useful fields.** If the input is SQL, use the SQL editor to limit the fields coming in.
* **Filter first.** Place a FILTER tool immediately after the SELECT reducing your data stream further.
* **Limit new spatial objects.** Be careful with spatial processing and make sure that you are not creating objects that you don't need. They are large and can slow things down.
* Eliminate unnecessary columns in joins.
* Don't process unneeded whitespace.
* **Update the size of the string data types to be real-world** not the default. US state codes are a string(2), leaving it at V\_String(254) is wasteful. This saves space and processing time by explicitly stating the size, less work for the engine. The “Auto Field” tool can automate this process.
* **Appropriately size your columns**. Use the autofieldstrings – sometimes columns are unnecessarily wide
* **Be number savvy.** Use a FORMULA tool to convert strings to numbers if you'll be doing math on them downstream. Number data types are conservative with their space usage. Also, doing this all upfront makes for a cleaner canvas as you're not having to add FORMULA tools before a SUMMARIZE, for example.
* **No doodling.** Remove BROWSE DATA and BROWSE MAP tools. Rendering that data can be expensive. Once a module is ready for prime time, disable the browse tools. On the advanced tab select "disable all browse tools". Disable all browse tools before automating the process.
* **Plan your algorithm first!** Then, start actually placing tools, and configuring those tools to do what the module should do, after a general plan is available.

## New Workflow Definition

On creating any new workflow, begin by naming the workflow with a descriptive title which is as short and succinct as possible. Naming standard should be of the form:

[Org Abbreviation]-[Area]-[Project]-[Intent]
e.g., CO-Fin-MonthlyClose-Create\_Final\_Summaries.yxmz

Also, include the same descriptive title (at least the [Intent] portion) as a comment box at a larger font level within the workflow itself.



Update the Metadata for that workflow on the Meta Info Tab One of the workflow options before configuring anything is to put in the name of the workflow, author, company, copyright, etc. There is also a place to put a description for the purpose of the workflow. While there isn’t a macro or hot-key to put this in, it is most likely a good place to start.

1. Author of Workflow:
2. Company:
3. Date created:
4. Data Sources:
5. Purpose:
6. Other Notes:

This effort to fill the description with meaningful information is especially useful if uploaded to the Gallery because this information displays without the user having to download the workflow.

Optional: The URL field, if filled, can link out to further documentation or to the department or other key home page if available.

Additionally, if uploading to the common Gallery, include a descriptive icon to improve visual cues.

## Tool-Level Best Practices

When placing tools on the canvas the following best practices are encouraged.

* Rename tools when building complicated wizards. When renaming tools, use a standard”
	+ Select: sel\_
	+ File: fil\_
	+ Join: join\_
	+ Multi-Join: mjoin\_
	+ Formula: for\_
	+ Web developers should mark the tools updated through the wizard interface.
	+ Wizard input descriptions should closely match the value names, and the order they are written in the plugin code should match the order in the test wizard. We debug a wizard problem by loading the test wizard with data from the error report, and if the names are out of synch, and/or in different orders, it can be difficult to figure out what goes where. This greatly helps with debugging.
* Use relative paths for datasets. Avoid hard coded values of datasets if/when possible. As data changes think of ways a module or wizard can accommodate changing data or sets of data without having to reconfigure tools each time. It can be time consuming to track down problems when wizards are integrated into sites and down the road data sets are changed. Also, mixing relative and non relative paths can be confusing.
* Update tool values with full replace or formula. Whenever possible, refrain from doing a "replace specific text" option as often the tool will be reconfigured in the UI and that value may no longer exist.
* Try to keep conditional statements/flow in the wizard update/action properties, not within the tool update formula itself – this just makes it a bit easier to see the update flow more quickly.
* Give good descriptions for each wizard input and update action – dealing with multiple tools and updates that have no description is a bit tedious.
* Provide default wizard input values so the user has at least a general idea of what needs to be submitted.

Other Tool Guidelines

* Rename spatial objects to what they are so that tools downstream can be configured properly.
* When performing a spatial match (pt in polygon) have the polygons specified as the Target
* Use the dynamic input tool for spatial matching
* Use yxdb files as input rather than XLS or other flat file options
* If the input is SQL, use a SQL login not a network login
* When setting up large data sets (data1 to data2) type fuzzy matches, it saves a ton of time and processing if you do a join upstream on the fields you will be doing the fuzzy matching on and eliminate them from the fuzzy match because they were exact.

## Tool Alignment

When a substantial number of tools are placed on the canvas they should be aligned and spaced in a way to create maximum readability including minimizing distracting clutter of connections where possible.

* **Alignment.** Individuals tools within a workflow can be aligned either horizontally (Ctrl Shift -) or vertically (Ctrl Shift +). This is the simplest change to make the workflow easy to review. Organizing the tools gives the impression of simple step by step process which is a value add for handover to business users or clients. The best practice is also to limit the number of backward connections as much as possible and use straight lines connecting tools whenever possible. When aligning the tools, you can either keep the tools on the same line or you can keep the connections straight. Both choices lead to a visually consistent look.
* **Spacing.** Allow no less than one or two tool widths between two consecutive steps. Distance also depends on whether an annotation is used or not. If so, allow more space to let the user read it easily and navigate faster.
* **Wireless Connections.** Tools far from each other but which are still logically related directly should be connected without a visible wire. Long lines across the workflow bring confusion and are unnecessarily giving a sense of overcomplexity. The wireless connection can only be either on the input or the output of the tool, not necessarily both. This is especially effective where there are in workflow “checks” and summarizations which are essentially off-to-the-side. Connecting the inputs to these calculations via wireless, removes the clutter of the long connector wire across the canvas and typically through other tools.

# Workflow Documentation

Both internal (within the workflow) and external (separate Word document) should be completed for any workflow which will be shared, reused or re-run with any consistency. Note that “1-time” workflows tend to be the most permanently reused items so try to document everything as much as possible even if the goal ends up to remind yourself in the future.

## Internal

There are a number of ways to improve the readability of the workflow. The Comment boxes are the single simplest approach. Some additional general guidelines include the following.

* Document everything in the module using text tools.
* Comments should be written for someone who is not familiar with the module, not for someone who is familiar with it.
* Put a simple text line with your name and date, sometimes with a comment as to what was revised. This is a simple text box with the borders turned off, shoved into an out of the way corner.
* Separate data processes and reporting processes in the module so you can quickly improve and trouble shoot.
* Name connections coming into reporting tools. This allows easier organization of report snippets and helps with the legend in mapping.

Comments documentation boxes should use a consistent color coding to guide the eye and intent as follows:



## Container Usage

Containers should be used to visually separate the workflow into component parts. This is very effective in helping quickly understand the “big picture” of a complex workflow. Color coding should largely follow the Comment box usage above. Be sure to use descriptive headers sometimes including numbers, e.g., (Step 6: Add Airline Ticket Number to SAP data)



## External

Each workflow should have an associated high-level documentation overview which is updated with *every* revision of the workflow. This should be the equivalent of a 1-pager analytic output. The goal is to define the purpose, inputs, outputs and key business rules of that workflow for consumption by people either without tool access or as a general framework reference and orientation before opening the workflow.

The general document structure should be as follows.

It should include:

* Title
* Requestor Name and Requestor Organization
* Workflow Author
* Version number and short description of changes since last version
* Short Description of the business Problem and Intent
* Any detailed business requirements in short bullet point form
* Screen shots or links to output and conclusion

This is mainly intended to be a handover document and tracker and should be limited to 1-page. Documentation should be mainly inside the workflow. This supplementary document is mainly intended for people who don’t work in or around the tool with any regularity.

It should not be:

* A complete documentation of the flow
* A complete list of business requirements
* More than 1-page

# Standard Macros

The following lists the key company standard macros. Remember don't reinvent the wheel: Build a Macro when creating a process that will be used on multiple occasions.

* Macro 1
* Macro 2
* Macro 3

# Unit Testing

Given the fact that it is so easy to make changes in Alteryx, it is even more likely to make changes which introduce errors. As such, time should be spent creating a follow-on workflow or set of tests to check the consistency of the output product. General approaches include the following.

* Write tests.
* Peer code review. Having someone else present your work in a group setting to review and have them be responsible to explain the why, what & how of the workflow will illustrate inconsistencies very quickly
* Use the event function and email. The email should have
	+ consistent subject lines in the automated emails so that Outlook can catalog them quickly
	+ text in the body of error messages that tells the user \*what\* to do, not just "it's broken"
	+ When working on an automated process, make sure you know what version of Alteryx is on both beta and live sites, and develop on the earliest version (or downgrade to the earliest version if necessary).

# Workflow Version Control

Workflows and dependent reference files like lookups should be checked into the version control repository frequently. Ideally use the built-in versioning of the Gallery for workflows. Additional repositories to store R and Python code could used but could also be injected via a script.

# Server / Gallery

The gallery for the organization serves as a central point for sharing, version control and automation for the organization.

## Location

Server location.

## Permissions

Overview of permissions related to the organization and how it was implemented in Gallery.

# Integration with External Tools

## Tableau

## R

## Python

# Contributors

Contributors are not listed in any specific order. Materials are drawn from the community and individual communications. Comments welcome especially errors or omissions of credit (it was unintentional).

| ***#*** | ***Contributor Name or Community Handle*** | ***Source / Community URL***  |
| --- | --- | --- |
| 1. | Paula Eldridge ([peldridge](https://community.alteryx.com/t5/user/viewprofilepage/user-id/147)) | <https://community.alteryx.com/t5/Engine-Works-Blog/Alteryx-Best-Practices/ba-p/1921>  |
| 2. | [Paultno](https://community.alteryx.com/t5/user/viewprofilepage/user-id/25290), [ADerbak](https://community.alteryx.com/t5/user/viewprofilepage/user-id/14865) | <https://community.alteryx.com/t5/Alteryx-Designer-Discussions/Alteryx-Workflow-Style-Guidelines/td-p/171718>  |
| 3.  | Ben Moss, The Information Lab | https://www.theinformationlab.co.uk/2019/02/14/documentation-best-practices-with-alteryx/  |
| 4. | 3danim8 (aka Ken Black)  | <https://3danim8.wordpress.com/2016/07/01/alteryx-strategies-for-solving-tough-problems/> |
| 5. | (Multiple – including Ken\_Black’s attachment further in the thread) | https://community.alteryx.com/t5/Community-Lounge/Do-you-have-an-organizational-or-personal-way-of-documenting/td-p/44768 |
| 6.  | Michal | <https://community.alteryx.com/t5/Alteryx-Designer-Discussions/Tips-and-Tricks-How-to-make-your-Alteryx-Workflow-look-nicer/td-p/152429> |
| 7. | DavidM | <https://community.alteryx.com/t5/Alteryx-Designer-Discussions/Alteryx-vs-Version-Control-Code-Repo-Git-R-and-Python-code-Best/m-p/430815> |

# Version History

Document version history.

| ***Version #*** | ***Date*** | **Author** | **Summary of Changes** |
| --- | --- | --- | --- |
| 0.1 | 7/15/2019 | [Hayes Williams](https://www.linkedin.com/in/hayeswilliams/) ([willhaye](https://community.alteryx.com/t5/user/viewprofilepage/user-id/56334)) | Initial compilation of various community articles and blog posts. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |