***TABLE OF CONTENTS***

[1 Introduction 3](#_Toc15634875)

[1.1 Overview and Purpose 3](#_Toc15634876)

[1.2 Definitions and Acronyms 3](#_Toc15634877)

[2 Environment 3](#_Toc15634878)

[2.1 User Folder Structure 3](#_Toc15634879)

[2.2 Alteryx User Environment Guidelines 3](#_Toc15634880)

[3 Workflow Development Best Practices 4](#_Toc15634881)

[3.1 Guiding Principles 4](#_Toc15634882)

[3.2 New Workflow Definition 5](#_Toc15634883)

[3.3 Tool-Level Best Practices 6](#_Toc15634884)

[3.4 Tool Alignment 7](#_Toc15634885)

[4 Workflow Documentation 7](#_Toc15634886)

[4.1 Internal 7](#_Toc15634887)

[4.2 Container Usage 8](#_Toc15634888)

[4.3 External 8](#_Toc15634889)

[5 Standard Macros 9](#_Toc15634890)

[6 Unit Testing 9](#_Toc15634891)

[6.1 Detect and Raise Error Conditions Options 10](#_Toc15634892)

[7 Workflow Version Control 10](#_Toc15634893)

[8 Server / Gallery 10](#_Toc15634894)

[8.1 Location 10](#_Toc15634895)

[8.2 Permissions 10](#_Toc15634896)

[8.3 Email Event Notifications for Workflows 10](#_Toc15634897)

[9 Workflow Migration 11](#_Toc15634898)

[9.1 Config File for the Environment 12](#_Toc15634899)

[9.2 Database Aliases 12](#_Toc15634900)

[9.3 Automated Migration 12](#_Toc15634901)

[10 Integration with External Tools 13](#_Toc15634902)

[10.1 Python 13](#_Toc15634903)

[10.2 R 13](#_Toc15634904)

[10.3 Tableau 14](#_Toc15634905)

[11 Contributors 14](#_Toc15634906)

[12 Version History 16](#_Toc15634907)

# 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 several 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.

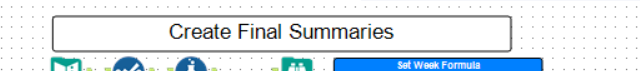
* **Don't handle data you're not going to process.** Place a SELECT tool immediately after your INPUT limiting your data stream to the fields that you are concerned with. Add descriptions. Even if you are not currently limiting your data stream, adding this tool now could save you headaches down the 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

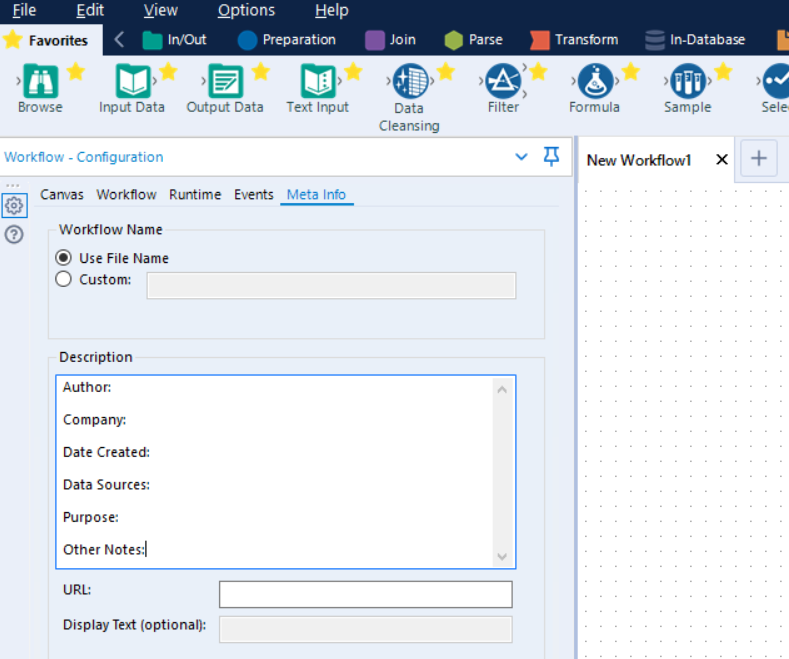
On creating any new workflow, begin by naming the workflow with a descriptive title which is as short and succinct as possible. Naming standard should be of the form:

[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

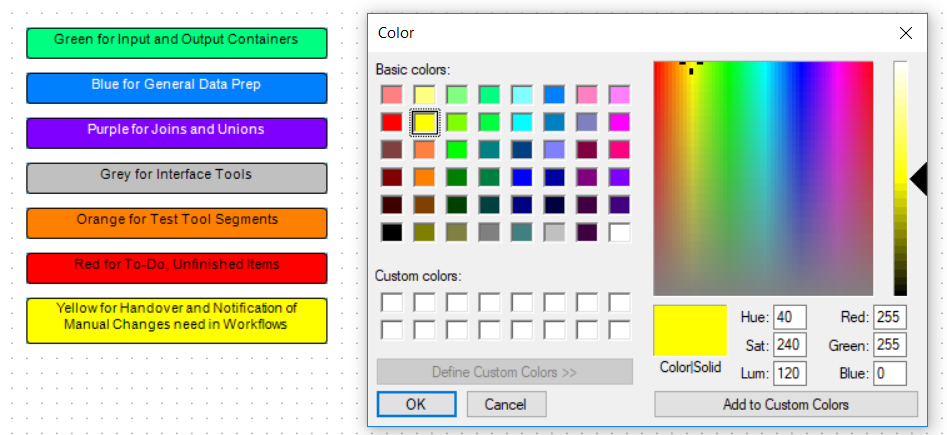
Both internal (within the workflow) and external (separate Word document) should be completed for any workflow which will be shared, reused or re-run with any consistency. Note that “1-time” workflows tend to be the most permanently reused items so try to document everything as much as possible even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

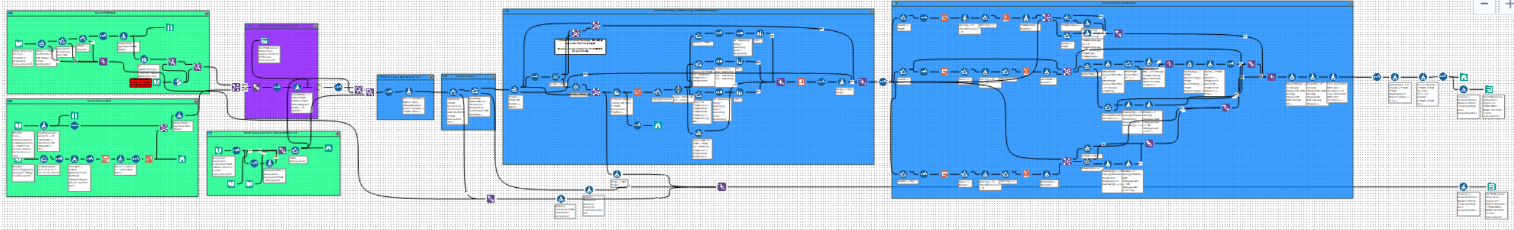
* Document everything in the module using text tools.
* Comments should be written for someone who is not familiar with the module, not for someone who is familiar with it.
* Put a simple text line with your name and date, sometimes with a comment as to what was revised. This is a simple text box with 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).

## Detect and Raise Error Conditions Options

I order to make it clear when things go wrong in a workflow, there are a number of approaches to detecting and raising error conditions as follows.

After doing some exploring, it seems that there are four different ways to detect and raise error conditions in workflows:

Methods in which execution continues even with error:

1. Test tool
2. Message tool with "Error" set as the message type

Methods to stop execution with error:

1. Message tool with "Error - And Stop Passing Records Through This Tool" set as the message type
2. An Error Message tool wrapped inside of a macro

# 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

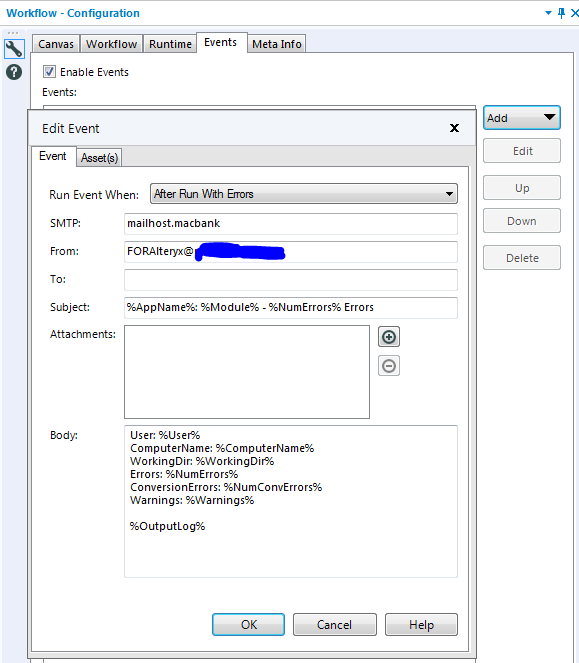
Overview of permissions related to the organization and how it was implemented in Gallery.

## Email Event Notifications for Workflows

For workflows which are being distributed to business users, it is very useful to enable an email event upon completion with details on the run to an email mailbox. Include key metadata (shown next) in the email to help make supporting the workflow with the business quicker.

* %AppName%
* %Module%
* %NumErrors%
* %User%
* %ComputerName%
* %WorkingDir%
* %NumErrors%
* %NumConvErrors%
* %Warnings%
* %OutputLog%

This is enabled with the “Run Event When:” toggle at the top of the option screen. It can be set to “After Run With Errors” so the problem runs have details immediately available.



# Workflow Migration

While making changes in a visual programming environment are quick and easy from a structured change management and development perspective the down side is that changes in a visual programming environment are quick and easy. For production tasks with business-critical outcomes a structured migration process should be followed like best practices in coding. Workflows should be moved from a development to a test to a production environment based on structured testing and review. In order to make this process as simple as possible and reduce the potential for errors, the following additional steps should be followed.

## Config File for the Environment

Use a config file with same naming and have identical folder structures in each environment for development, test and prod. The config file contains all parameters that are environment specific. Each environment will have a separate config file tailored to that environment.

Use relative referencing when pointing to the configuration file, e.g., (..). See Options, Advanced Options, Workflow Dependencies -> All Relative.

When moving between environments, copy and paste all workflows and macros without any changes. The relative referencing will pick up the relevant locations from the new environment.

Macros must NOT be mapped to the Alteryx toolbar (Options, User Settings, Edit User Settings, Macros) because they will no longer be referenceable using the relative referencing method. They actually disappear from the Workflow Dependencies entirely. Only have the macros mapped to the Alteryx toolbar in the Production environment. To schedule your workflows on server you need any macros to be in the Alteryx toolbar in the Production environment. Relative referencing doesn't work when scheduling from disk. Need to verify this in the new version.

## Database Aliases

The methods Andrew has suggested are a great idea. I would also advise to use Alias Manager (Options->Advanced Options->Manage DB Connections) for any DB connections as you can for instance have an Alias called 'DataWarehouse' which points to your Dev DW when on your Dev Alteryx Servers and your Prod DW when on your Prod DW Servers.

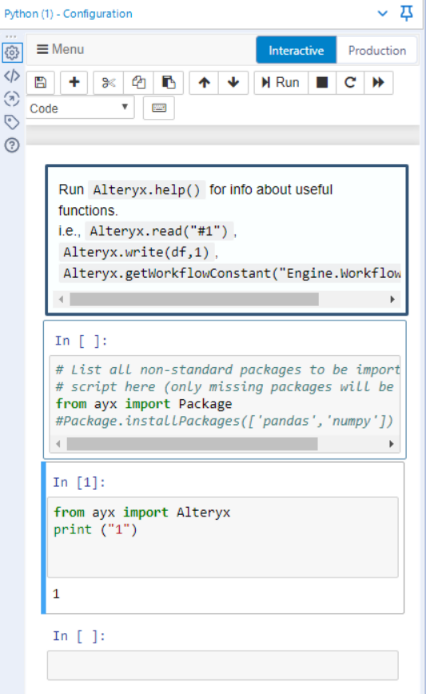
Also, if you need to pass connection\directory parameters into a Macro, you can use a text box that is configured as a question which will create a workflow constant, which you can then reference the input with %Question.TextBox%\file.csv

## Automated Migration

With 2018.4, the ability to do migration of workflows within the server environment has been enabled. This is located within the Admin portal to enable workflow migrations. This requires some setup by the administrator since it is enabled with a set of API endpoints between source and target gallery environments.

# Integration with External Tools

## Python

Integration with Python code will likely be necessary where the tools required for processing have not been directly integrated into the Alteryx palette as of yet. This will likely be required in a good portion of more complex data science analyses. The Python tool is available (as of 2018.3) to integrate external Python code into workflows via Jupyter Notebooks.

The notebook takes 2-3 seconds to initially instantiate after dropping the Python tool onto the canvas so some patience is required before the notebook session displays.

Data can be brought into the tool via the anchor but it is not required.

The following libraries are available by default:

* Ayx (Alteryx API)
* Jupyter
* Matplotlib
* numPy
* pandas
* requests
* scikit-learn
* scipy
* six
* SQLAlchemy
* Statsmodels

Additional packages can be installed as required using the Alteryx.installPackages() in Python tool.

Python must be installed on the computer with Alteryx for this to be active either natively or via Anaconda or similar system.

## R

Like the Python tool, the R tool is available to extend the available functionality to include native R code. Designer reads valid R scripts and passes the scripts through the tool. Output from the R tool matches the R outputs.

To aid in debugging, R scripts within the R tool, add print statements to the code. Output from these statements will appear in the message log. Also, many issues arise from incorrect Object Classes alignment. Alteryx can read data as a list or data frame and attempts to write as a data frame. Most data should be coerced into a data frame to avoid these conflicts.

Additional R packages can be installed using the “[Install R Packages](https://gallery.alteryx.com/#!app/Install-R-Packages/57bb2a58a248970b4472c2e6)” app in the “Predictive District” of the Gallery.

R must be installed on the system for this tool to display and work effectively as a prerequisite to usage.

## Tableau

# 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> |
| 8. | chriseHe | Python   * [All about the Python tool [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_Tool-2DMastery-2DPython_ta-2Dp_197860&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=UfTmhNhDyXHlRGysIu8OMIm69Y2HFmvfn62Roswo42Q&e=) * [Python libraries included [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_Python-2DTool-2DLibraries-2DAn-2DIntroduction-2Dto-2DPython_ta-2Dp_193781&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=5129jbaz7uH7jLw3Aj54mauq5UUkwTTXJRQUynTjC4A&e=) * [How to install new packages [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_How-2DTo-2DUse-2DAlteryx-2DinstallPackages-2Din-2DPython-2Dtool_ta-2Dp_406244&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=XeImliu6AUllIz-5U8gKj_WDG3-6TUKsxuaQJEzIteU&e=)   R   * [Help doc has a great overview [help.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__help.alteryx.com_current_R.htm&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=DZTe39G6CvoVZAP631cBRsm1bRsE_FfQW-VbzbBLTdQ&e=) * [R tool tips and tricks [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_R-2DTool-2DTips-2Dand-2DTricks-2DPart-2D1_ta-2Dp_6579&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=r3fJh830eRws3RgRWEk6XcAwdbJp02BPAUA16y_gbFE&e=) * [Install custom packages [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_Install-2DR-2DPackages_ta-2Dp_41265&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=t4w_mpKSPZFu84VJuJ65FKnJLqOLSWEpXVwePZOnfzE&e=) * [R packages with Alteryx [community.alteryx.com]](https://urldefense.proofpoint.com/v2/url?u=https-3A__community.alteryx.com_t5_Alteryx-2DDesigner-2DKnowledge-2DBase_R-2DPackages-2Din-2Dthe-2DR-2DTool_ta-2Dp_256177&d=DwMFAg&c=AgWC6Nl7Slwpc9jE7UoQH1_Cvyci3SsTNfdLP4V1RCg&r=HJsrF_g58jSUZVnd4xTBUZ1zWMT2o6CerAMkCjmfiJ8&m=R3L4Q-1N7UJgEVGAIdVJ4yAOncNGWEe9NnxWkVUz848&s=mbnktWn6_DYHxm0aWms3gKA9Alfx1eK9ca-tEvtM0nU&e=) |
| 9. | ChrisTX | Error detection   * Add an email event (under the Events tab);Run Event When: After Run with Errors <https://community.alteryx.com/t5/Inspire-2019-Buzz/Give-us-Your-Best-Tips-and-Tricks-Calling-all-Tip-Meisters-for/td-p/387312/page/2> * Methods to detect and raise error conditions: <https://community.alteryx.com/t5/Alteryx-Designer-Ideas/Consistent-Error-Behavior-for-Test-Message-Tools/idi-p/169695>   Workflow Migration   * Best Practice: Workflow configuration for migration to multiple environments (dev / prod / etc) <https://community.alteryx.com/t5/Alteryx-Designer-Discussions/Best-Practice-Workflow-configuration-for-environment/m-p/19475#M8660> |
| 10. | JeffreyP | Automated Migration with 2018.4 <https://community.alteryx.com/t5/Alteryx-Server-Knowledge-Base/Migrating-Workflows/ta-p/335774> |

# 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. |
| 0.2 | 8/1/2019 | Inclusion of changes from the Chris’es: [@chriseHe](https://community.alteryx.com/t5/user/viewprofilepage/user-id/34663) and [@ChrisTX](https://community.alteryx.com/t5/user/viewprofilepage/user-id/44249) | Content added to:   * R & Python sections * Addition of Error Detection and Migration sections |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |