Every Expert Was Once a Novice: Getting Started with Autodesk® PLM 360

Rodney Coffey – Razorleaf Corporation

PL2429

Autodesk® PLM 360 offers a powerful set of tools that that can be used to configure solutions to a customer's business needs. In this class, we discuss best practices for getting started in Autodesk® PLM 360. We cover how to properly manage your Autodesk® PLM 360 project using the tool itself. This includes technical methods and examples for environment change management and roll out. The goal of this class is to enable administrators to be efficient, from planning to deployment.

Learning Objectives

At the end of this class, you will be able to:

- Describe Best Practices for Getting Started in PLM 360
- Be prepared for scripting on top of your solutions later
- Recognize the importance of managing best Practices and Environment Change
- Recognize that with great flexibility in configuration, comes great responsibility

About the Speaker

Rodney is a PLM Consultant for Razorleaf Corporation. He has spent the last 10 years of his career in the Autodesk Manufacturing Channel responsible for Training, Implementation, Migration, and Solution Building. He has extensive experience around the Autodesk manufacturing products including Autodesk AutoCAD, Inventor, Vault, and PLM 360. In his current role his primary focus is the PLM 360 product. He and his team have partnered with Autodesk Consulting to provide customers with a better understanding of how PLM 360 can solve their business problems. These services include 360 scripting, training, implementation, and development.

Rodney.Coffey@razorleaf.com

Razorleaf Corporation

Introduction

Why Best Practices are Important

Regardless of the software being implemented or the size of the project, a few key points are common:

1. End Users and Administrators are looking for tools that give them freedom to design and configure solutions to their Problems

2. Once they have found the right tool, they will quickly begin to limit methods and practices that others should use when using that tool

While this sounds somewhat contradictory, often times the companies that do this best, early and often in an implementation project for any software are those that are most successful. Generating best practices around the tools and processes that we follow is critical and provides many of the following benefits:

- Helps define the way a tool should be used for a given company
- Significantly reduces the learning curve of a product or new administrators
- Keeps environment configurations uniform and scalable
- Ensures that things are done because they *should* be, and not just because they *can* be
- Creates efficiency across an implementation team
- Adds conformity to an implementation team trying to reach common goals

Throughout this session we will look at a number of simple best practices that have been learned and used within Autodesk® PLM 360. Many of the practices that we will look at are simple but can have a great impact throughout the course of an implementation project. It is important to note that while best practices are to be enforced, they may change over time due to product enhancements, or changes to your business process.

Configuration Naming Best Practices

Whether a complex part numbering system that tells everything we need to know about the item it represents or just a plain sequential numbering system; Naming Conventions are important. A well thought out naming system configured in Autodesk® PLM 360 will increase efficiency in the following areas:

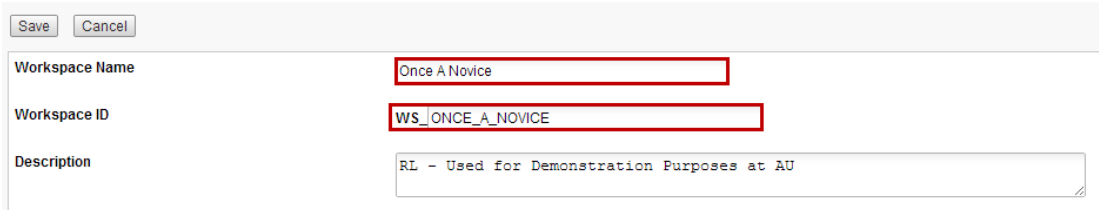
- Find / Search / Sort on Data
- Write Scripts using Workspace / Field IDs
- Manage the environment over a length of time (Today, Tomorrow, 2 Years from now)
- Effectively manage your implementation project and team
- Separate standard App, and Autodesk® PLM 360 content from what you are using in production

Workspaces

Best Practice: Workspace Name = Workspace ID

When creating a workspace in Autodesk® PLM 360 a workspace is given a name by the user. Autodesk® PLM 360 uses this name to generate a Workspace ID. Once the workspace is created, the name may be changed, but the Workspace ID may not. All internal references to the workspace made via scripting or the API are made through the Workspace ID. As such, adequate planning before Workspaces are named will save countless headaches for the admin and developers of your team.

- To make things Manageable keep these the same



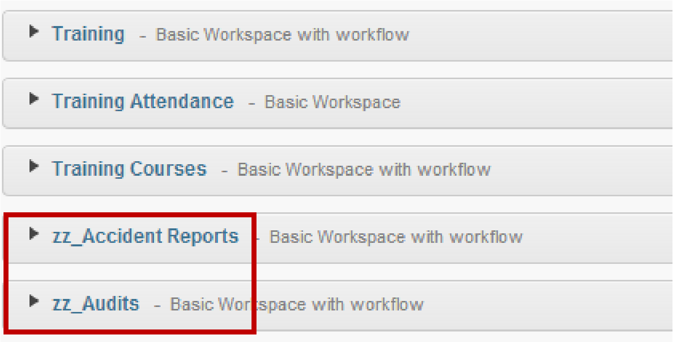
Workspace Name	Once A Novice
Workspace ID	WS_ONCE_A_NOVICE
Description	RL - Used for Demonstration Purposes at AU

- If a workspace is given a name that needs to be changed use the *Clone* Workspace Feature to create a new Workspace with the correct name. This will ensure the two remained in Sync. The workspace with the incorrect ID can then be deleted. Note that Workflows are not cloned upon cloning a workspace, but can be copied and pasted from the workflow editor of one workspace to another.

Clone

Best Practice: Workspace Name = Workspace ID – Exception to the Rule

When we receive our Autodesk® PLM 360 tenants they come preloaded with valuable applications that are pre-built to suit many of our business needs. We may also leverage the Autodesk® PLM 360 App store to further tailor our tenant to meet the needs of our business processes. You may find that Workspaces developed during a testing or preview phase have merit, but aren't ready to be rolled out to users. Rather than deleting them, these workspaces may be tucked away for future use. When working within the Workspace manager all the workspaces show up in Alphabetical order whether they're being used or not. This can make navigating the list cumbersome, and deleting whole workspaces and all their associated data is not always the right answer. Prefixing the Workspaces with a common set of characters (zz_ will push them to the bottom) will separate them from production workspaces until you are prepared to work with them.



▸ Training - Basic Workspace with workflow
▸ Training Attendance - Basic Workspace
▸ Training Courses - Basic Workspace with workflow
▸ zz_Accident Reports - Basic Workspace with workflow
▸ zz_Audits - Basic Workspace with workflow

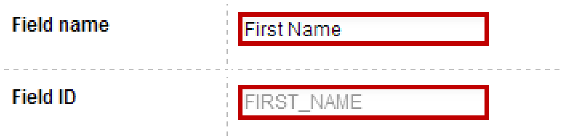
Fields

Best Practice: Field Name = Field ID

When creating workspace fields, Autodesk® PLM 360 assigns internal IDs, much like the workspaces discussed above. The initial internal ID is based on the name the user gives the field during creation. The issues discussed above are all relevant to Fields as well. The importance of this best practice, however is somewhat higher. While we may have a handful of workspaces that are relevant to an Autodesk® PLM 360 application, we likely have hundreds of fields. Often times, scripts are built and are repurposed to work with multiple workspaces. As a result, a mis-named field will result in lost time much like a mis-named Workspace. Minutes cloning a field to give it the appropriate name and Matching ID can save countless time hunting for the correct ID for use in some other development activities. If one is planning an Integration using the Autodesk® PLM 360 API or Jitterbit this can quickly complicate data mapping, and will likely impact the time table for a successful integration project. Below is a list of Autodesk® PLM 360 common solution elements that can be affected by the poor implementation of this best practice.

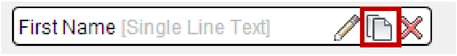
- Use of Computed Fields
- Scripting
- Integration
- API Development

- To make things Manageable keep these the same



Field name	First Name
Field ID	FIRST_NAME

- If a workspace field is given a name that needs to be changed use the *Clone* Feature to create a new field with the correct name

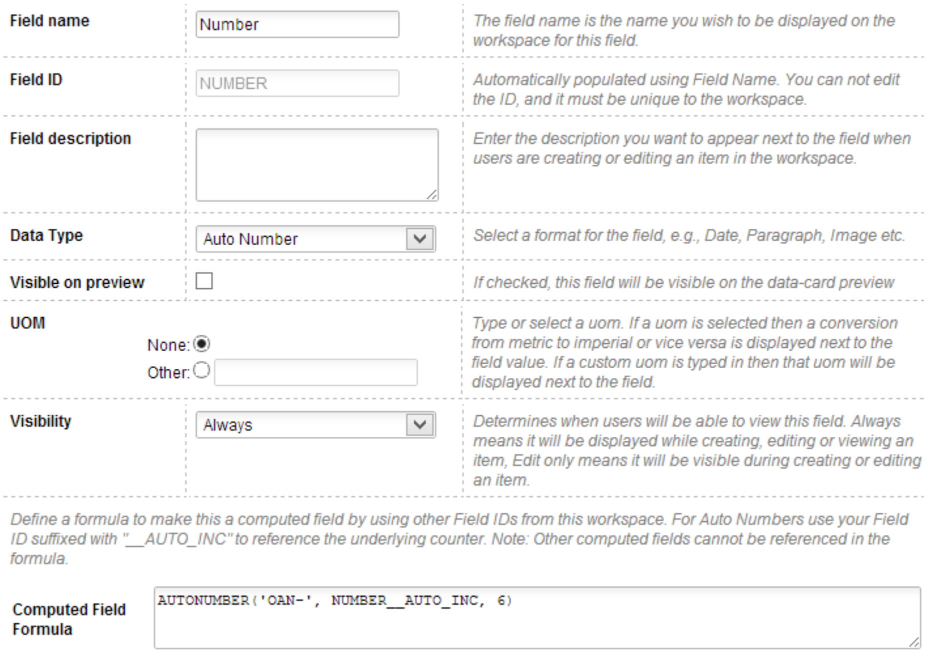


First Name [Single Line Text]

Best Practice: Create a Unique Identifier

As you begin to use your workspace in a Production environment you will be generating hundreds of records. Although Autodesk® PLM 360 maintains uniqueness by an object's dmsID, users will want a more significant way to recognize their items as unique. This may be done through a combination of fields in a descriptor or by applying an Auto-Number field to the workspace as a unique Identifier. Sometimes it is appropriate to use both methods. This unique identifier is displayed to the end users of the system in the Item browser via your choice in the workspaces descriptor. This is how you will display the unique identifier to the end users.

- Below an Auto Number field is set up to do Sequential Numbering in the workspace. This will ensure each Item in the Workspace has a Unique Number for easy identification by the end users



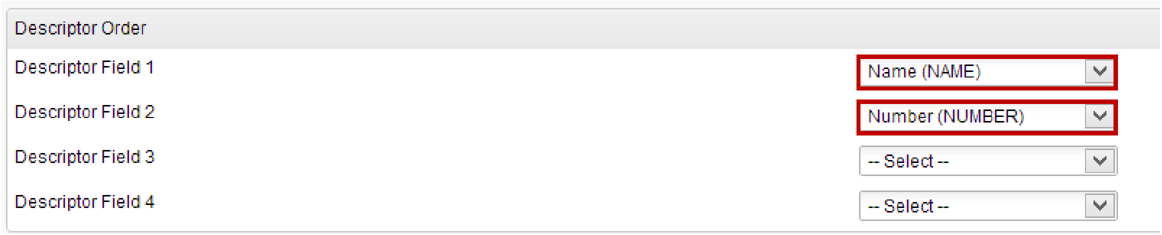
Field name	Number	*The field name is the name you wish to be displayed on the workspace for this field.*
Field ID	NUMBER	*Automatically populated using Field Name. You can not edit the ID, and it must be unique to the workspace.*
Field description		*Enter the description you want to appear next to the field when users are creating or editing an item in the workspace.*
Data Type	Auto Number	*Select a format for the field, e.g., Date, Paragraph, Image etc.*
Visible on preview	□	*If checked, this field will be visible on the data-card preview*
UOM	None: ● Other: ○	*Type or select a uom. If a uom is selected then a conversion from metric to imperial or vice versa is displayed next to the field value. If a custom uom is typed in then that uom will be displayed next to the field.*
Visibility	Always	*Determines when users will be able to view this field. Always means it will be displayed while creating, editing or viewing an item, Edit only means it will be visible during creating or editing an item.*

Define a formula to make this a computed field by using other Field IDs from this workspace. For Auto Numbers use your Field ID suffixed with "__AUTO_INC" to reference the underlying counter. Note: Other computed fields cannot be referenced in the formula.

Computed Field Formula

```
AUTONUMBER('OAN-', NUMBER__AUTO_INC, 6)
```

- Below a Descriptor is setup to Uniquely identify items



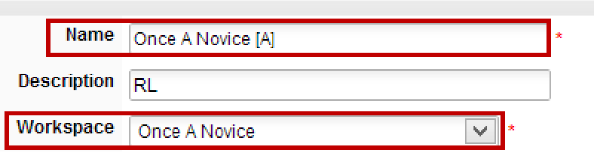
Descriptor Order	
Descriptor Field 1	Name (NAME)
Descriptor Field 2	Number (NUMBER)
Descriptor Field 3	-- Select --
Descriptor Field 4	-- Select --

Roles / Groups

When a workspace is created in Autodesk® PLM 360 it is necessary to create Roles with Permissions to actually grant users' access to use the workspace. An administrator will have the need to create multiple Roles depending on the access that is to be provided to the workspace. These Roles are then applied to Groups. Obviously as we scale our solutions and provide access to many users within an organization access becomes critical and we need to be able to quickly identify how access is defined.

Best Practice: Role Names

A role name should always look like the below, including its general usage and the workspace for which it is used.



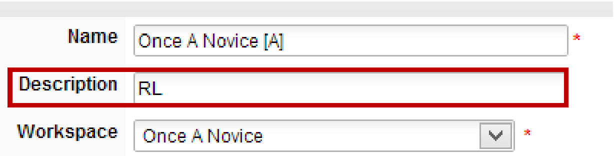
Name	Once A Novice [A] *
Description	RL
Workspace	Once A Novice *

The general usage of a Role can be broken down by the below suffix abbreviation which will for the most part be found in the "Out of the Cloud" solution, and should be used when configuring your own solutions.

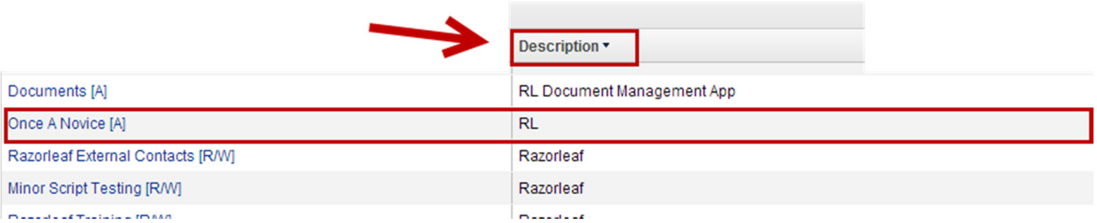
- [R] – Read Only Users (No Edit Capabilities – Consider Participant Licenses)
- [R/W] – Read / Write Users (Add and Edit but limited capability to Delete)
- [A] – Administrative Users – All Access to a given workspace and its capabilities
- [Permission Name – WF] – Workflow related Roles

Best Practice: Role Description

Include your company name / acronym in the description. This comes in handy later on as the lines between prebuilt workspace roles and your production used roles become blurry. This is the easiest way to sort the Roles later on.



Name	Once A Novice [A] *
Description	RL
Workspace	Once A Novice *



	Description ▾
Documents [A]	RL Document Management App
Once A Novice [A]	RL
Razorleaf External Contacts [R/W]	Razorleaf
Minor Script Testing [R/W]	Razorleaf

Guideline / Reference: Roles Applied to Groups

Roles will be applied to Groups to grant access to users. It is important to plan this accordingly, and also name groups in a method Administrators will understand. Below is an example that mimics a Vault Role and Group Structure.

PLM Role Name Reference	PLM Group Name Reference	Vault Role / Group Name Reference
Once A Novice [R]	Once A Novice Consumers	Consumers
Once A Novice [R/W]	Once A Novice Editors	Editors
Once A Novice [A]	Once A Novice Admins	Administrators

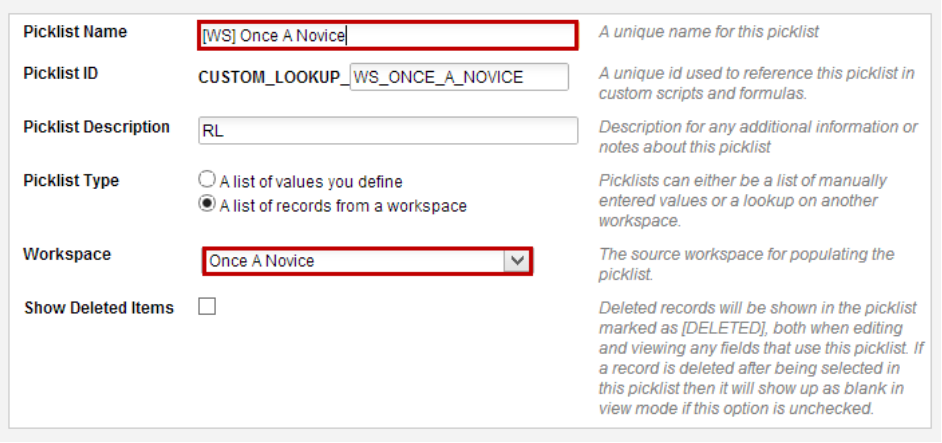
As a general rule choose these names carefully and plan ahead. If you have other systems already in place to organize your users into common groups, use those.

Picklists

Best Practice: Picklist Naming

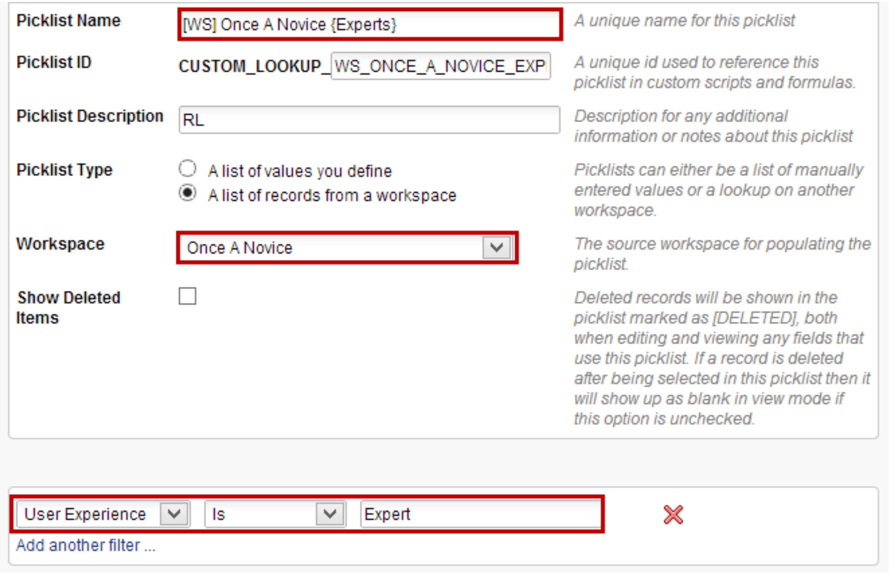
Picklists like other features created in PLM should be craftily named so that Users can easily identify their purpose and use. Usually, common sense names are adequate enough. Keep in mind pick lists can be shared amongst many workspaces. There are a few unique cases where the name should follow a best practice. These are illustrated below:

- When a pick list is created based on a Workspace a user should prefix the pick list as shown below. This will help identify that list type from the other user-created types.
  - [WS] Workspace - based Picklist



Picklist Name	[WS] Once A Novice	A unique name for this picklist
Picklist ID	CUSTOM_LOOKUP_ WS_ONCE_A_NOVICE	A unique id used to reference this picklist in custom scripts and formulas.
Picklist Description	RL	Description for any additional information or notes about this picklist
Picklist Type	○ A list of values you define ● A list of records from a workspace	Picklists can either be a list of manually entered values or a lookup on another workspace.
Workspace	Once A Novice	The source workspace for populating the picklist.
Show Deleted Items	□	Deleted records will be shown in the picklist marked as [DELETED], both when editing and viewing any fields that use this picklist. If a record is deleted after being selected in this picklist then it will show up as blank in view mode if this option is unchecked.

- When a filter is applied to a picklist based on a workspace, a user should add a qualifier to the end of the picklist name as shown below.
  - [WS] Workspace-based picklist {Qualifier}



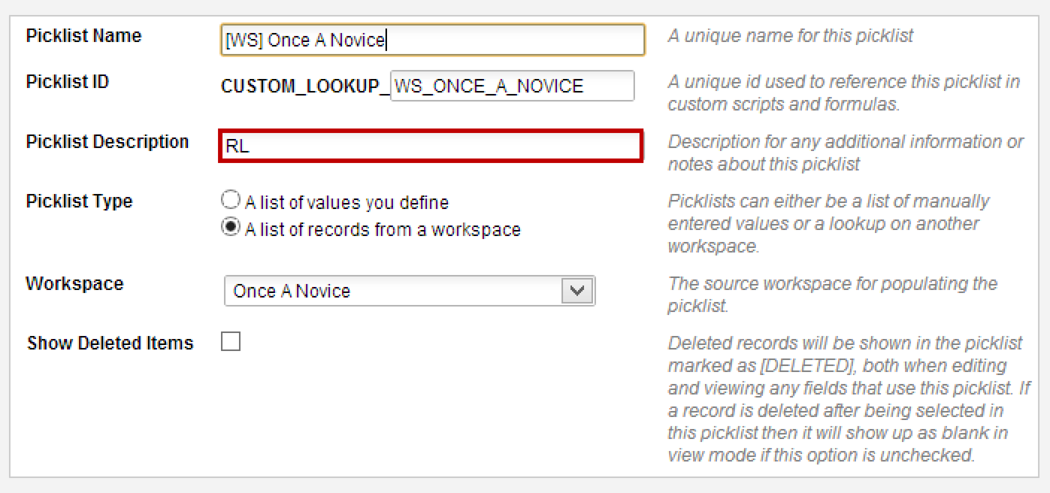
Picklist Name	[WS] Once A Novice {Experts}	A unique name for this picklist
Picklist ID	CUSTOM_LOOKUP_ WS_ONCE_A_NOVICE_EXP	A unique id used to reference this picklist in custom scripts and formulas.
Picklist Description	RL	Description for any additional information or notes about this picklist
Picklist Type	○ A list of values you define ● A list of records from a workspace	Picklists can either be a list of manually entered values or a lookup on another workspace.
Workspace	Once A Novice	The source workspace for populating the picklist.
Show Deleted Items	□	Deleted records will be shown in the picklist marked as [DELETED], both when editing and viewing any fields that use this picklist. If a record is deleted after being selected in this picklist then it will show up as blank in view mode if this option is unchecked.

User Experience | Is | Expert ✕

Add another filter ...

Best Practice: Picklist Description

Include your company name / acronym in the description. This comes in handy later on as the lines between prebuilt pick lists and your production used pick lists becomes blurry. This is the easiest way to sort the picklists later on.



Picklist Name	[WS] Once A Novice	A unique name for this picklist
Picklist ID	CUSTOM_LOOKUP_ WS_ONCE_A_NOVICE	A unique id used to reference this picklist in custom scripts and formulas.
Picklist Description	RL	Description for any additional information or notes about this picklist
Picklist Type	○ A list of values you define ● A list of records from a workspace	Picklists can either be a list of manually entered values or a lookup on another workspace.
Workspace	Once A Novice	The source workspace for populating the picklist.
Show Deleted Items	□	Deleted records will be shown in the picklist marked as [DELETED], both when editing and viewing any fields that use this picklist. If a record is deleted after being selected in this picklist then it will show up as blank in view mode if this option is unchecked.

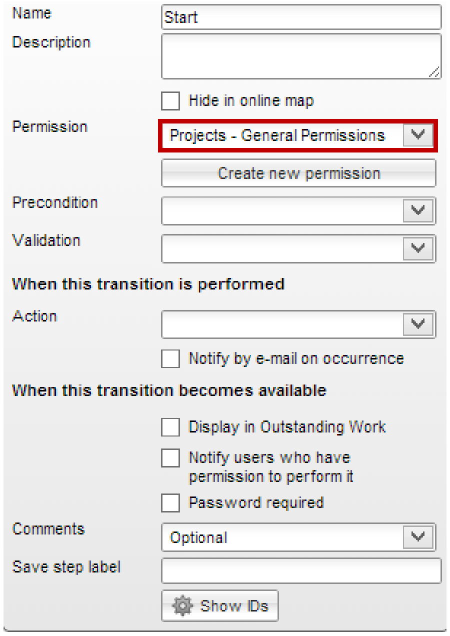
Workflow Best Practices

When creating a workflow in Autodesk® PLM 360 there are a number of best practices that should be considered by the user or administrator.

Best Practice: Permission Naming Conventions

When naming Permissions use the Workspace Name or acronym to which it applies and the general method of the permission

- Projects – General Permissions



Name: Start

Description:

□ Hide in online map

Permission: Projects - General Permissions

Precondition:

Validation:

When this transition is performed

Action:

□ Notify by e-mail on occurrence

When this transition becomes available

□ Display in Outstanding Work

□ Notify users who have permission to perform it

□ Password required

Comments: Optional

Save step label:

Creating Permissions in a workflow

When a user generates a workflow in the workflow editor they are required to add permissions to that workflow in order to enable access to users via the assigning of the permissions to Roles, and then Groups.

1 2 3

Permission > Role > Group

1 2 3

Projects – General Permissions > Once A Novice [General – WF] > Once a Novice Editors

Best Practice: Permission Creation – Keep it Simple

Don't create permissions just because you can. It is easy to see the flexibility of the tool and begin to think of all the different avenues you can create in your workflow. It is easy to begin creating a permission per transition. When this usually catches up to us is when we go to apply the permissions to Roles, and then effectively assigning those Roles to groups. As a practice work from the other direction.

- Create a *General Permissions* permission
- Use this through your early testing and development
- Consider Validation scripting to block transitions that have requirements to be filled
    - Fields Required, Approvals, Tasks Completed, Ownership
- Then go back and carefully assess where additional workflow permissions are required
    - Below is a list of Common areas where you'll probably need additional permissions
        - Migration Workflows
        - Restricting or Giving groups of People Reject / Cancel Permissions
        - Block Hidden Transitions and Workflow states from Workflow Actions

1. General Permissions Applied
2. Cancel Permissions Applied

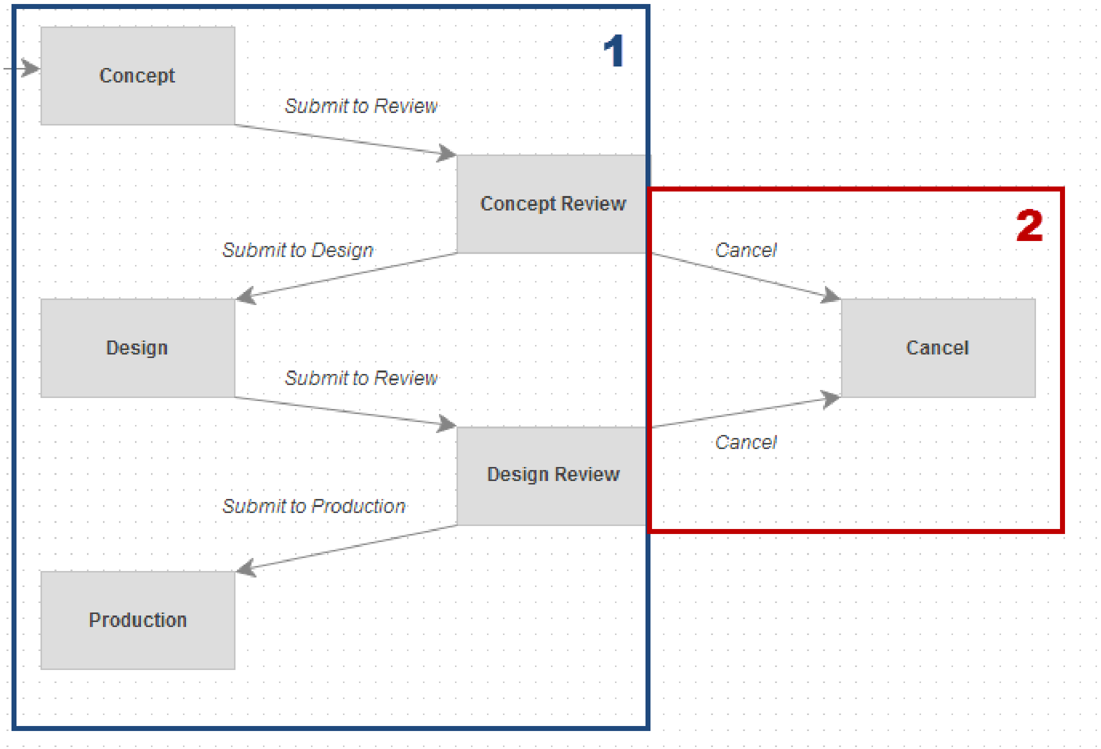


Figure 1 Core and Cancel Workflow Permissions

Assigning Permissions to their own Role

Permissions, Roles, and Groups are a common area that new administrators of Autodesk® PLM 360 get stuck until that moment when it all clicks. There is one thing that still gets missed and that is the assigning of Workflow permissions to a dedicated Role. This falls into that area as earlier discussed "Don't do it just because you can". When Workflow permissions find their way into other Roles (R, R/W, A) the clarity of the solution becomes blurred. Workflow permissions play a critical role in why a user can or cannot perform certain actions. For example a user without the correct workflow permissions cannot add rows to a grid, attachments, or obviously promote an item. It is much easier to troubleshoot such issues when each WF permission is assigned to its own role, and then that role is assigned to the appropriate groups.

Best Practice: Permissions are applied to a dedicated Role

Each workflow Permission is assigned to its own Role it so it can be easily assigned and managed.

Note the below example of the two Groups and their permissions. We want User 1 to have Cancellation Rights, and User 2 have all rights except Cancellation

Reference: Permission > Role > Group>Users

Projects – General Permissions > Projects [R/W] > Project Editors > User 1, User 2

- Both can read / write Item Details, Grid, Attachments, etc.

Projects – General Permissions > Projects [General – WF] > Project Editors > User 1, User 2

- User 2 can move the Item through the Core workflow (see section 1 of Figure 1 above), but cannot cancel

Projects – Cancel Permissions > Projects [Cancel– WF] > Project Cancellation > User 1

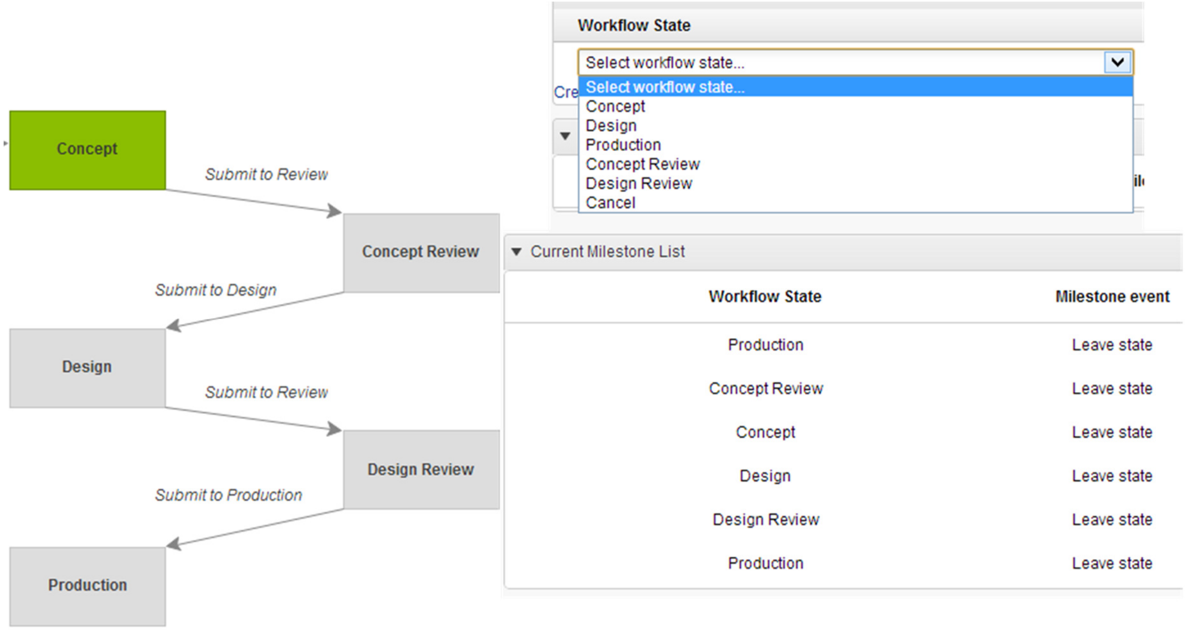
- User 1 can move the Item through the Cancel workflow (see section 2 of Figure 1 above), and can also move the item through the Core workflow because they are also in the Project Editors Group

Using Numbers in your Workflow (Milestone Related)

When creating a workflow that will later be used with Milestones, Project Management, or scripting, it is important to know the order in which the workflow states are going to be reached. This is not apparent when building a workflow because you see the states in the workflow editor in the order you built them. When the workflow states are later used to create Milestones for an Item, it is not the order they were created in but the date that is scheduled on the Milestone that is the key piece of information for ordering Milestones.

Best Practice: Assign a Number to your Workflow states to insure the correct order of events is used later on in Milestones

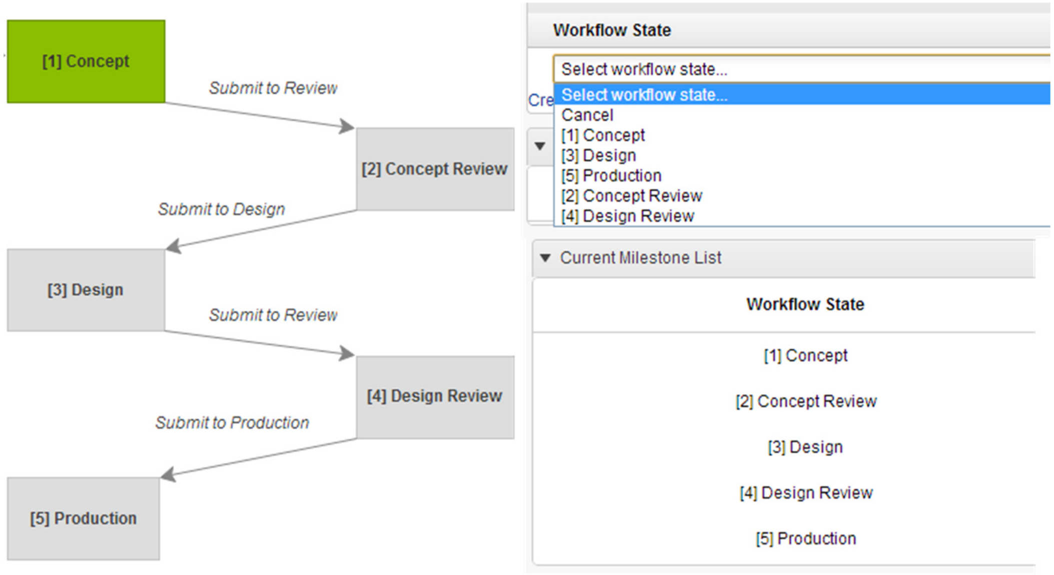
In Figure 2 below, note the Workflow on the left and the Milestones applied to the item on the right. The Milestones are out of order, but that is not easily identified.



Workflow State	Milestone event
Production	Leave state
Concept Review	Leave state
Concept	Leave state
Design	Leave state
Design Review	Leave state
Production	Leave state

Figure 2 Milestone List (Out of Order)

In Figure 3 below, again note the Workflow on the left and the milestones applied to the item on the right. The Workflow now has numbers (e.g., [1]) at the start of each workflow state name. These show up in the drop down list, and provide correct ordering of the Milestones in the list.



Workflow State
[1] Concept
[2] Concept Review
[3] Design
[4] Design Review
[5] Production

Figure 3 Milestone List (In Order)

General Scripting Best Practices

There are a number of things that are important to consider when developing scripts in Autodesk® PLM 360. Of course, a little scripting experience can go a long way when building custom solutions, but there are some things you may want to consider when it comes to managing your scripts and code.

Script Naming

As administrators and developers generate scripts in Autodesk® PLM 360 they need to easily be able to identify what scripts are for. Similar to the other areas discussed in this section we have devised a schema as a best practice for naming scripts. Something to consider is the amount of scripting or development that you may need for your Autodesk® PLM 360 environment. Scripting may be created and managed elsewhere (Visual Studio, Etc) but used in PLM. This makes the name somewhat critical so portability is easy. This also again helps distinguish what is Production used and prebuilt Out Of the Cloud. Below are best practices for scripting names.

Best Practice: Script Naming Conventions (Non-Library)

Script Name Break Down:

1 2 3 4
Company Name/Acronym _ Script Type _ Workspace/App _ Purpose

Example:

1 2 3 4
RL_A_Product_S

Script Type Acronyms:

- A = Action
- V = Validation
- C = Condition

Purpose Type Acronyms (Most Commonly Used Scripting Functions)

- S = Spawning
- CG = Copy Grid
- MS = Milestone Creation
- PM = Project Management (Schedules)
- AT = Auto Transition

*When a Script is written and does not fall into a function Category, a short name is assigned to that section of the name.

Example:

RL_A_Product_Approval

Best Practice: Script Naming Conventions (Library)

Script Name Break Down:

1 2 3
Company Name/Acronym _ Script Type _ Function

Example:

1 2 3
RL_L_SpawnItemAddtoPicklist

Script Type Acronyms:

- L = Library

Script Headers and Revision (Title Block for Your Scripts)

Certain information can be highly useful to include inside your scripts. This information includes the script name, purpose, author, and change history. If you choose to write or manage scripts outside of PLM 360 in another tool (for example, Visual Studio), having this information stored can greatly enhance the ease with which you can find the appropriate script for a task. Also, keeping a change history communicates what has been updated in the script over time which can help in troubleshooting later on.

Best Practice: Create a header that contains critical tracking information for your scripts

Note the example below:

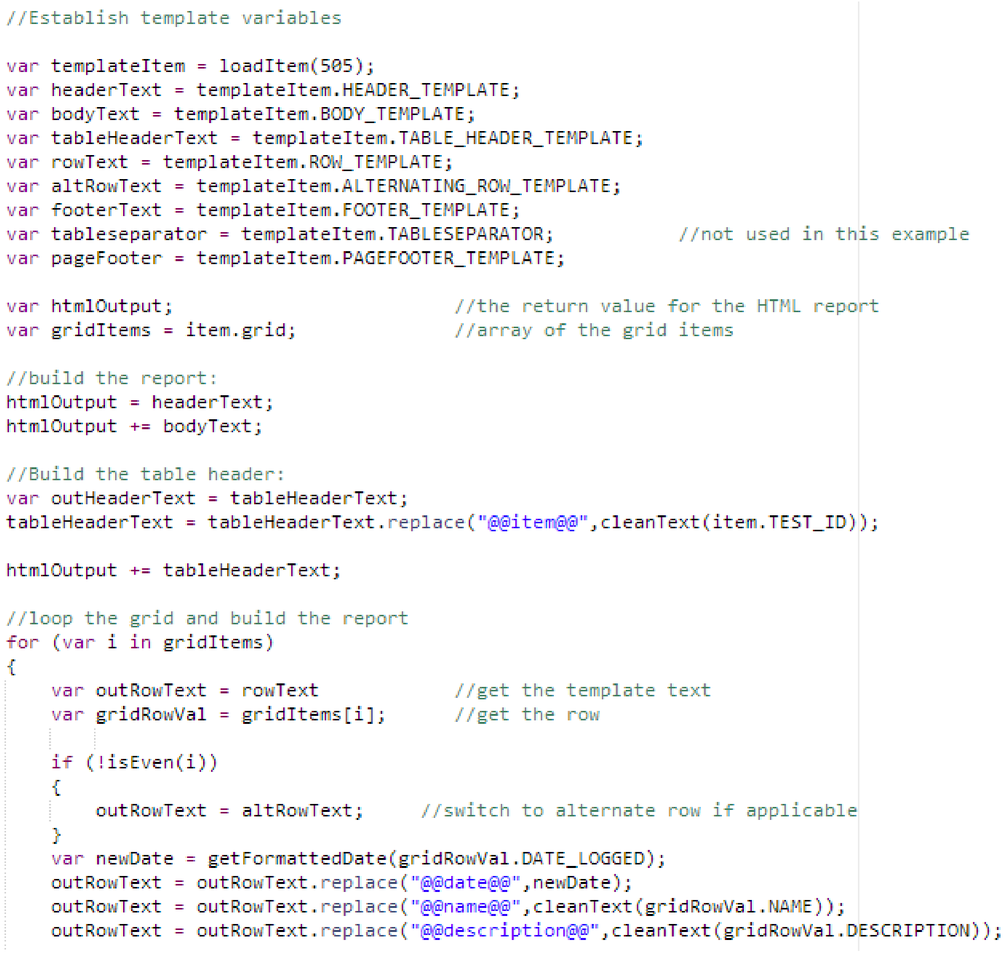
```
/****************************************************************************************
*****
Name: RL_A_Product_S
Description:
   This script was designed to Spawn all necessary documents required for a Product
Imports:
        RL_L_SpawnItemAddtoPicklist
Constraints:
        Fields:
        Documents: Associated Product
        Products: Spawned Document Fields; Fields must use a linking Picklist [WS] Documents

Revisions:
        Date                Modified By           Company           Description of Change
        ==========          ==============        =========         ======================
        2013-12-5                RAC              Razorleaf             Initial Creation
*****************************************************************************************
****/
```

Comment Scripts

Commenting scripts with notes as you build your scripts is important and should be considered a best practice. Although it can be a tedious practice, and takes a little extra time, below are some common reasons this is important.

- Revisiting a script for further development after a long period of time has passed
- Developing scripts in a group of administrators
- Further developing scripts created by others

Best Practice: Comment scripts to capture important instructions and thoughts during the script design process



```
//Establish template variables

var templateItem = loadItem(505);
var headerText = templateItem.HEADER_TEMPLATE;
var bodyText = templateItem.BODY_TEMPLATE;
var tableHeaderText = templateItem.TABLE_HEADER_TEMPLATE;
var rowText = templateItem.ROW_TEMPLATE;
var altRowText = templateItem.ALTERNATING_ROW_TEMPLATE;
var footerText = templateItem.FOOTER_TEMPLATE;
var tableseparator = templateItem.TABLESEPARATOR;          //not used in this example
var pageFooter = templateItem.PAGEFOOTER_TEMPLATE;

var htmlOutput;                         //the return value for the HTML report
var gridItems = item.grid;              //array of the grid items

//build the report:
htmlOutput = headerText;
htmlOutput += bodyText;

//Build the table header:
var outHeaderText = tableHeaderText;
tableHeaderText = tableHeaderText.replace("@@item@@",cleanText(item.TEST_ID));

htmlOutput += tableHeaderText;

//loop the grid and build the report
for (var i in gridItems)
{
    var outRowText = rowText            //get the template text
    var gridRowVal = gridItems[i];      //get the row

    if (!isEven(i))
    {
        outRowText = altRowText;     //switch to alternate row if applicable
    }
    var newDate = getFormattedDate(gridRowVal.DATE_LOGGED);
    outRowText = outRowText.replace("@@date@@",newDate);
    outRowText = outRowText.replace("@@name@@",cleanText(gridRowVal.NAME));
    outRowText = outRowText.replace("@@description@@",cleanText(gridRowVal.DESCRIPTION));
```

Best Practice: Use smart variable names, and although commenting is good, don't overdo it

Always use variable names that make sense to avoid the need for unnecessary commenting on simple operations. Comments should only need to be used to describe more complex logic that may not be easily understood just from reading it through.

Library Scripts

Library scripts are one of the four different types of scripts that can be developed inside of Autodesk® PLM 360. Library scripts contain functions that are then used in Action, Condition, and Validation scripts. Using library scripts for commonly used functions greatly reduces repetitive scripting, consolidates scripting efforts, and creates consistency across a development team. It is possible to import multiple library scripts into a single Action, Condition, or Validation script. Additionally, library scripts can be imported into other library scripts. This is an important practice used by the solution developers when building an application in Autodesk® PLM 360. Below are some common library scripts containing functions for sending email, adding a specific number of days to a date, and comparing the difference between two dates.

Best Practice: Create Libraries for commonly used functions

Examples:

- RL_L_sendEmail

```
    function sendEmail(to, subject, body) {
var emailMe = new Email();
emailMe.to = to;
emailMe.subject = subject;
emailMe.body = body;
emailMe.send();
}
```

- RL_L_addDaysToDate
  - Description: Used to Add Days to the current Date an item is created; Used commonly when creating Milestones, Schedule

```
    function addDaysToDate (startDate, daysToAdd) {
var futureDate = new Date();
var dayOfMonth = startDate.getDate();
futureDate.setDate(dayOfMonth+daysToAdd);
return futureDate;
}
```

- RL_L_compareDatesInDays
    - Description: Compares Two Dates from a workspace and returns the difference in Days. Used commonly in tracking the time between Tasks, Actions, or even workflow states.

```
     function compareDatesInDays(d1, d2) {
  var oneDayInMS = 1000*60*60*24;
  var d1_ms = d1.getTime();
  var d2_ms = d2.getTime();
  var diff_ms = d2_ms - d1_ms;
  var days = Math.round(diff_ms/oneDayInMS);
  return days;
}
```

Library Script Separation

It is best practice to create one Library script for a single function. When you avoid the urge to compile many functions into a common library, it greatly simplifies the management of the scripts, and aids troubleshooting efforts.

Best Practice: Break Libraries into separate scripts.

Library Script General Best Practice

Try to consider the solution as a whole when planning scripts. If there is common logic that will be used in several workspaces, put it in a library, or if it's being duplicated exactly, re-use the entire script across workspaces. If there is common logic used in various parts of a workflow, but perhaps reference different fields, use a switch statement based on the CustomTransID. This allows you to put all the actual logic in one place.

Best Practice: Plan ahead and Understand your entire solution before sitting down to automate

Implementation Best Practices

There are many other keys to success when working toward a successful implementation of PLM. These keys to success are not necessarily picks and clicks or yet another convention, but important process and planning items that can make all the difference in the road to a successful PLM implementation.

People Process Technology

Having good planning and good project management goes a long way towards keeping any implementation on the track to success. It is critical that we remember that our PLM implementation is about more than us implementing a new tool; it is also about how we will impact the people and process within our organizations. Carefully consider the below:

- People:
    - Who understands the process?
    - Who will be impacted by the solution?
    - Who should represent departments, divisions, or groups in an organization?
    - Who would give constructive feedback during early testing?
    - Who will be the PLM Champion(s)

- Process:
    - What is the current process?
    - Is the process already documented?
    - What is good about the current process?
    - What is broken in the current process?

- Technology:
    - Know the technology you're implementing
        - Get basic product training and understand the capabilities of the tool before diving into development on an application that will impact your organization

    - Consider the impact of new technology on old technology
        - What tools will be replaced?
        - What tools will be interfacing with your new tools?

    - Know and understand the software's capabilities to automate a Process.

    - PLM Technology is put in place to manage and drive a process. It still requires the people involved in the process to interact and understand their jobs. Even the best tools, and thorough process cannot guarantee success without the right people.

Map it, Build it, Use it…Map it, Build it, Use it…Go Live

Autodesk® PLM 360 offers powerful tools that allow an organization to quickly realize company-changing, process-enforcing solutions in a uniquely quick timeline. That being said; plan for plenty of testing throughout your development, as well as plenty of testing before deployment to your end-users. Remember that user acceptance testing (UAT) is not training, and training is not UAT. When rolling out your PLM 360 solution plan to focus your first build testing on a small group that understands the goals of the project well. Gather feedback from these sessions and apply changes as needed. Be sure to reengage this core group of users and get final approval on the finished state of the design before moving on to end user training. Depending on the solution and how many people it impacts you may do multiple iterations of this process.

Once the environment is finished (no more changes to be made!), then schedule end user training, when those users who are less familiar with the process and are learning how it should be completed will participate. Involve your core group of users in this training. It will greatly assist in your classes to have helpful hands; and these users will have a chance to see the finished product of their earlier testing, as well as increase their understanding of the finished solution and how it will be used.

- Map It
    - Fields that are to be captured by the solution
    - Process – Review or Build a Step by Step process map (Process Map is not equal to Workflow Map)
    - Workflow – Build the expected PLM workflow map (Make Notes of Scripting needs)
    - Consider Relationships to existing applications
    - Consider Data Migration – What needs to be loaded into the system in order for it to be in a Production State
    - Consider Data Dependencies – What other solutions must be built in order to deploy the one we are working on

- Build it
    - Plan your permissions and Roles
    - Build the solution (Workspaces, Item Details, Workflow, Security, etc.)
    - User Acceptance Testing – Test in a small team of administrators and core users that understand the overall goals of the project
    - Gather feedback from your core group of users in UAT and iterate the design and implementation of your solution
    - Prepare training materials for end users

- Use It
    - Training – Train End Users, and any other Administrators, in how to use the system
    - Go Live – Expect the Best, Prepare for the Worst. After all the testing and iterative design your solution should be prepared for roll out. While most times you will have hit the 90% mark, you must be prepared for your critics, and ready and able to support your roll-out. Allocate enough time and resources to be available when you launch your solution to the mass user base.

PLM for your PLM

Our PLM projects often have the same project and process management needs as the solutions we are developing inside Autodesk® PLM 360. A great way to get more familiar with the tool, and support your implementation efforts is to create solutions in your tenant to help manage your project and implementation efforts. Below are some applications to consider building from the start.

Project Management

Create a Project Management App that can be used to set the timeline of your implementation and its solution elements. You may have multiple items that represent different phases of your project. These can easily roll up to the master implementation Project. Consider the list below of things you would want to include.

- Item Details:
    - List Administrators
    - Team Members
    - Leads for different Solution Elements

- Workflow:
    - Phases of the project as you have them mapped out

- Project Management:
    - Add Tasks and Milestones

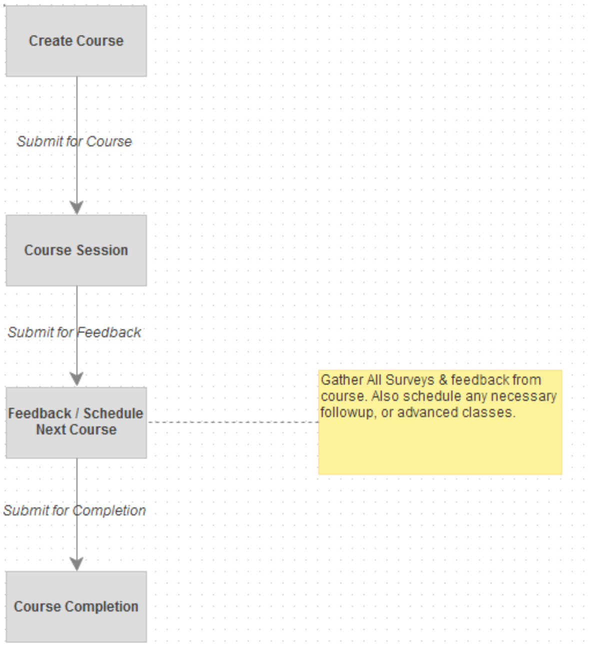
- Grid:
    - Tasks Management
    - Requirements Gathering

- Attachments:
    - Capture important Requirement Documents
    - Capture Sample data
    - Photos, Notes

Training

Create a Training App to help manage training efforts that will take place as part of your project. Consider the list below of things you would want to include.

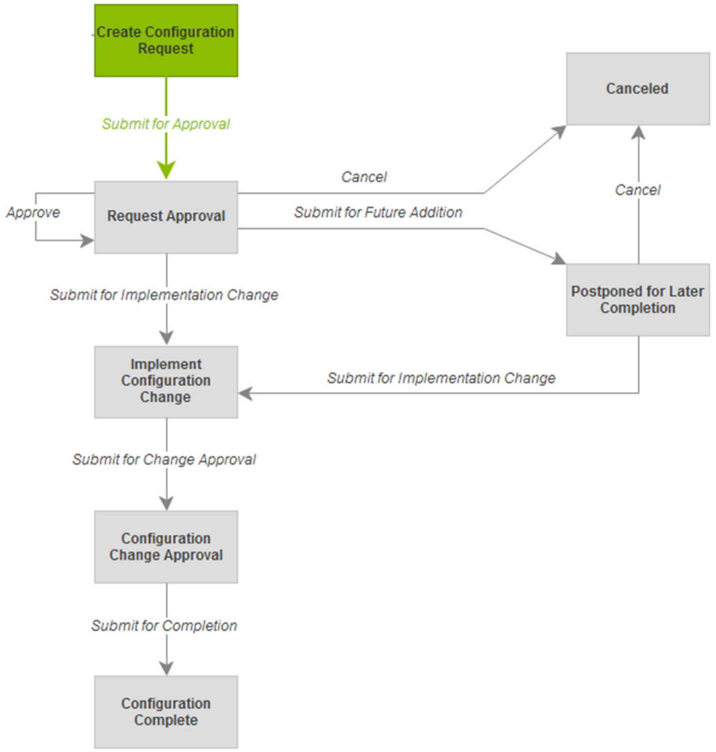
- Training Courses
    - Item Details:
        - Description
        - Length of Course
        - Pre-Requisites
        - Agenda
        - Periodic Refreshers
    - Attachments:
        - Exercises
        - Material
        - Presentations
- Training Sessions
    - Item Details (In Addition to the Above):
        - Invitees
        - Attendees
        - Scheduled Date
    - Relationships:
        - Training Courses


Create Course
Submit for Course
Course Session
Submit for Feedback
Feedback / Schedule Next Course
Gather All Surveys & feedback from course. Also schedule any necessary followup, or advanced classes.
Submit for Completion
Course Completion

PLM Configuration Change Management

As we build out and go live with the configuration of our Autodesk® PLM 360 environments there will be meaningful communication around how the environment could be changed or added to. This is common as roll-out begins and it is important to capture these thoughts, and give users an avenue to give feedback to the administrators about the system they have configured. Creating a Configuration Change Management workspace can greatly aid in managing these requests and feedback. Consider the list below of things you would want to include.

- Fields:
  - Related Application / Workspace
  - Type of Request: Change / Addition / Future State
  - Description
- Workflow:


Create Configuration Request
Submit for Approval
Request Approval
Approve
Cancel
Submit for Future Addition
Canceled
Cancel
Postponed for Later Completion
Submit for Implementation Change
Implement Configuration Change
Submit for Implementation Change
Submit for Change Approval
Configuration Change Approval
Submit for Completion
Configuration Complete