z/OSMF Hands-On Labs

- Choose Your Own Topic -

Create your own Workflow
(Workflow Editor)

Abstract:

Do you want to create your own z/OSMF Workflow? This lab will allow you to create your own z/OSMF Workflow using the z/OSMF Workflow Editor. The self-directed lab will show you how to add your own steps, variables, and jobs, and end up with your own Workflow that you can share with users!

The z/OSMF Workflow Editor is provided in z/OS V2.2 PTF UI42847 (closed January 3, 2017), and the z/OSMF V2.1 PTF UI43814 (closed February 1, 2017).

Some basic terms to get you started:

- **Workflow:**
    1. An activity associated with the z/OS system, such as configuring a component or product.
    2. The instantiation of a workflow in z/OSMF, based on a workflow definition. A workflow consists of one or more units of work to be performed on the z/OS system, as described by the workflow definition. A workflow is created when the Workflows task in z/OSMF is used to create an instance of a workflow from a supplied workflow definition file.
- **Workflow definition:** The logical structure of a workflow, represented through a series of steps, through the main XML file (the workflow definition file). The workflow definition identifies the various system objects and actions that constitute activities on z/OS and the rules for performing the activities.
- **Workflow variable input file:** An optional file that supplies default values for one or more of the input variables defined in the workflow definition file.
- **Step:** A single, logical unit of work in a workflow.

Exercise instructions

When you follow this self-directed lab, here is a high-level overview of what you will learn:

__ 1. With the z/OSMF Workflow Editor, create a Workflow from scratch.
__ 2. Import a step from the Step Library.
__ 3. Create steps in your Workflow.
__ 4. Define variables to be used in your Workflow.
__ 5. Provide defaults and variable specifications.
__ 6. Save your steps into a library for others to use.
__ 7. Retrieve steps from a library that others have provided.
__ 8. Save your Workflow.
__ 9. Try your Workflow out to make sure it is as you desire.
__ 10. Edit an existing Workflow to add, remove, and change steps.
__ 11. Try your Workflow out to make sure the changes are desirable.

Sample Workflow for this lab:

The Workflow that we will create in this lab is very simple. It will:

1. Create and run a step from the Step Library. (Using a REST API)
2. Run LISTCAT against a dataset. (Using JCL)

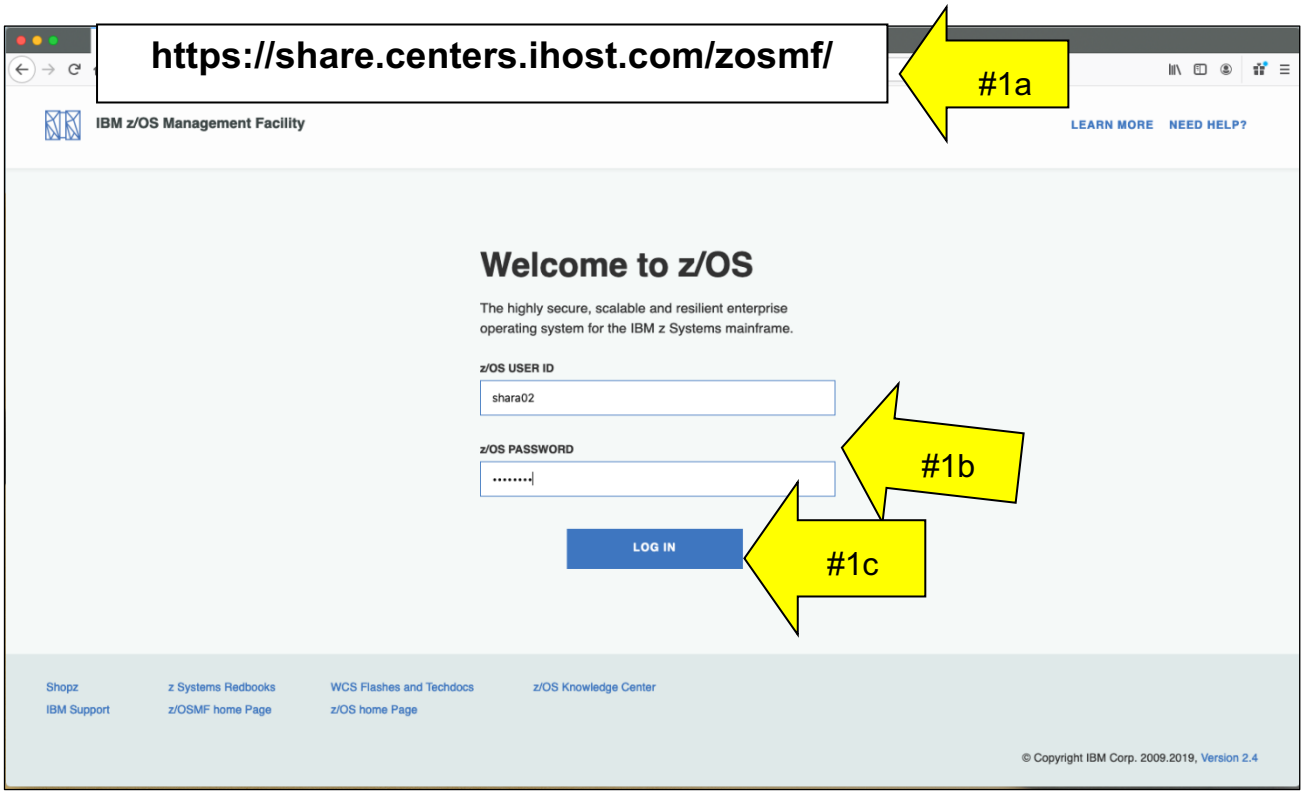
1. Logon to z/OSMF.

In this step, we will now go into z/OSMF to use the Workflow Editor function. For this lab, we are using a z/OSMF V2.5 system.

a. Go to https://share.centers.ihost.com/zosmf/ on the Firefox or IE web browser. (If you want to follow this lab on your own system, that is fine. Just note some of the samples we use you will need to supply yourself, using the Appendix to find those samples.).

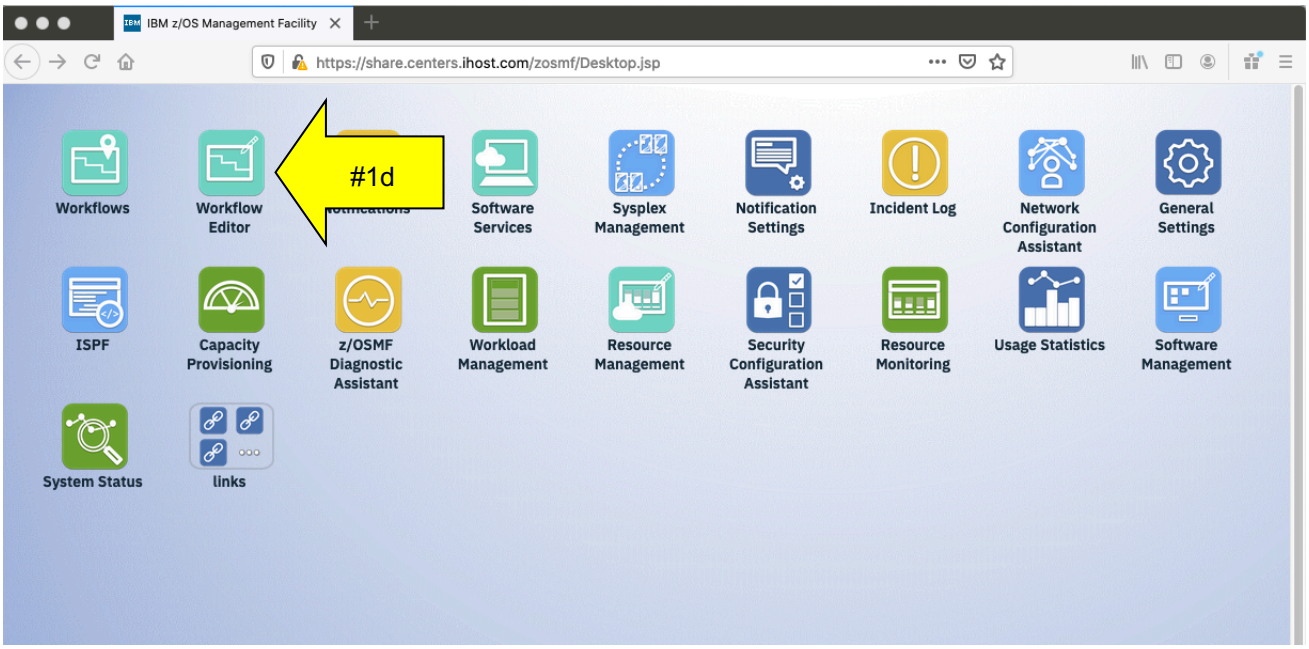
b. Using the userid you were given (SHARAnn, SHARBnn, or SHARCnn) and the password, logon to z/OSMF. The userid you were given is a regular z/OS userid on this system and has been given access to z/OSMF. There is *no* z/OSMF code on this workstation, all executables (except the web browser) is on the z/OS system.

c. Click on “LOG IN”.



Enter into the Workflow Editor function.

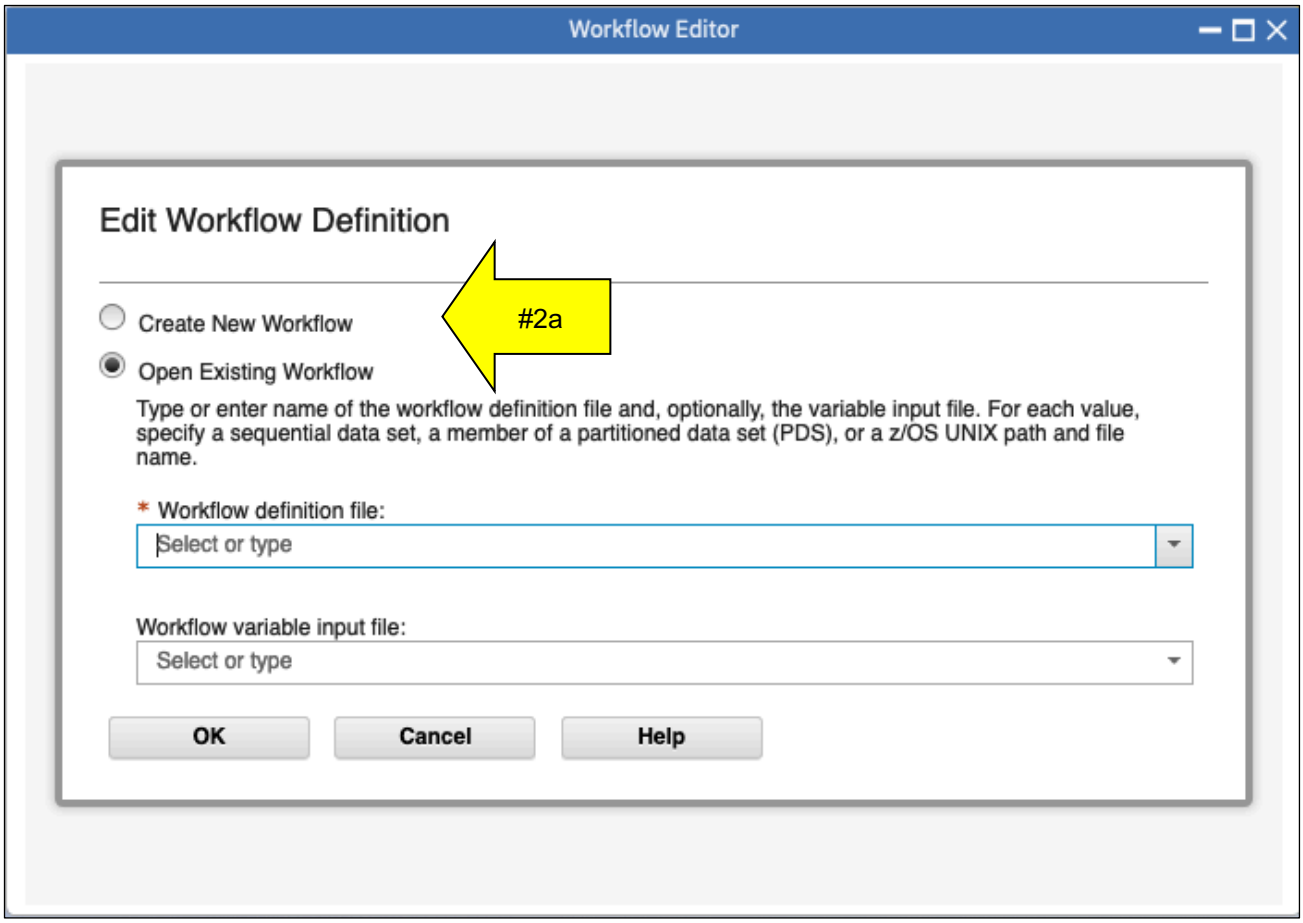
d. Click on “Workflow Editor”.



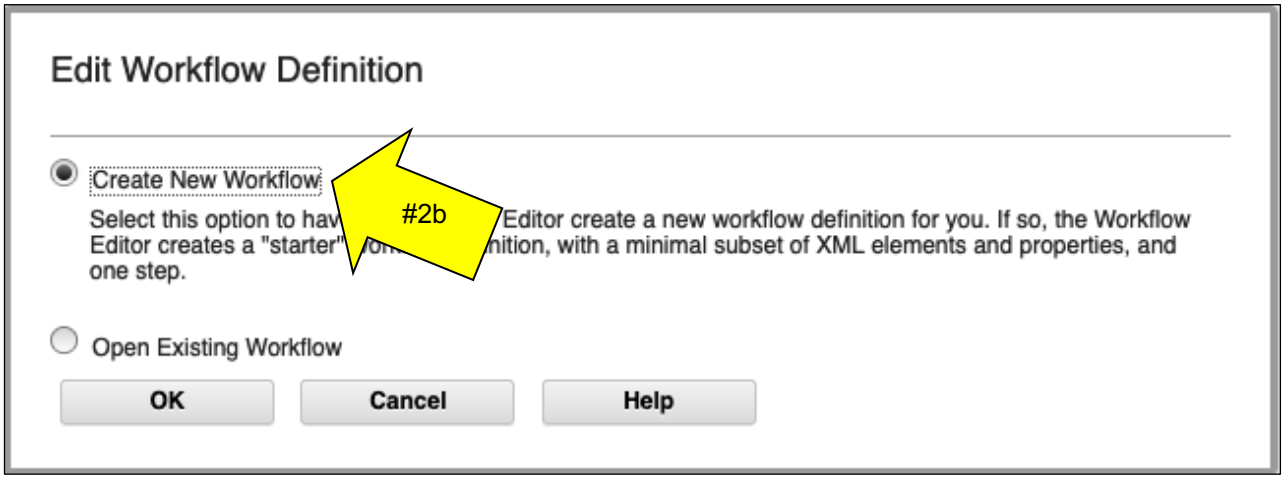
2. Start to create your new Workflow.

In this step, we will create a brand-new Workflow. You can see from this screen that you also could select an existing Workflow to edit. For now, we will create a Workflow from scratch.

a. Click on “Create New Workflow”. By default, you can see that “Open Existing Workflow” is selected. We’ll use that option later.



b. When you “Create New Workflow” this is the next dialog box that appears. Click on “OK”.



Metadata for your Workflow.

At this point, we need to provide some general information about our new Workflow, called “Metadata”, as indicated on the tab. There are other tabs for “Steps”, “Variables”, and “Input Properties”. You can see that there is even a scroll bar on the right, as we can’t even see all the information that we could provide here. The * fields are required. Make sure you scroll down to see more options.

At any time, you can click on the upper right “Help” to read more about each of these fields.

c. Workflow ID: type a meaningful name for this required field.

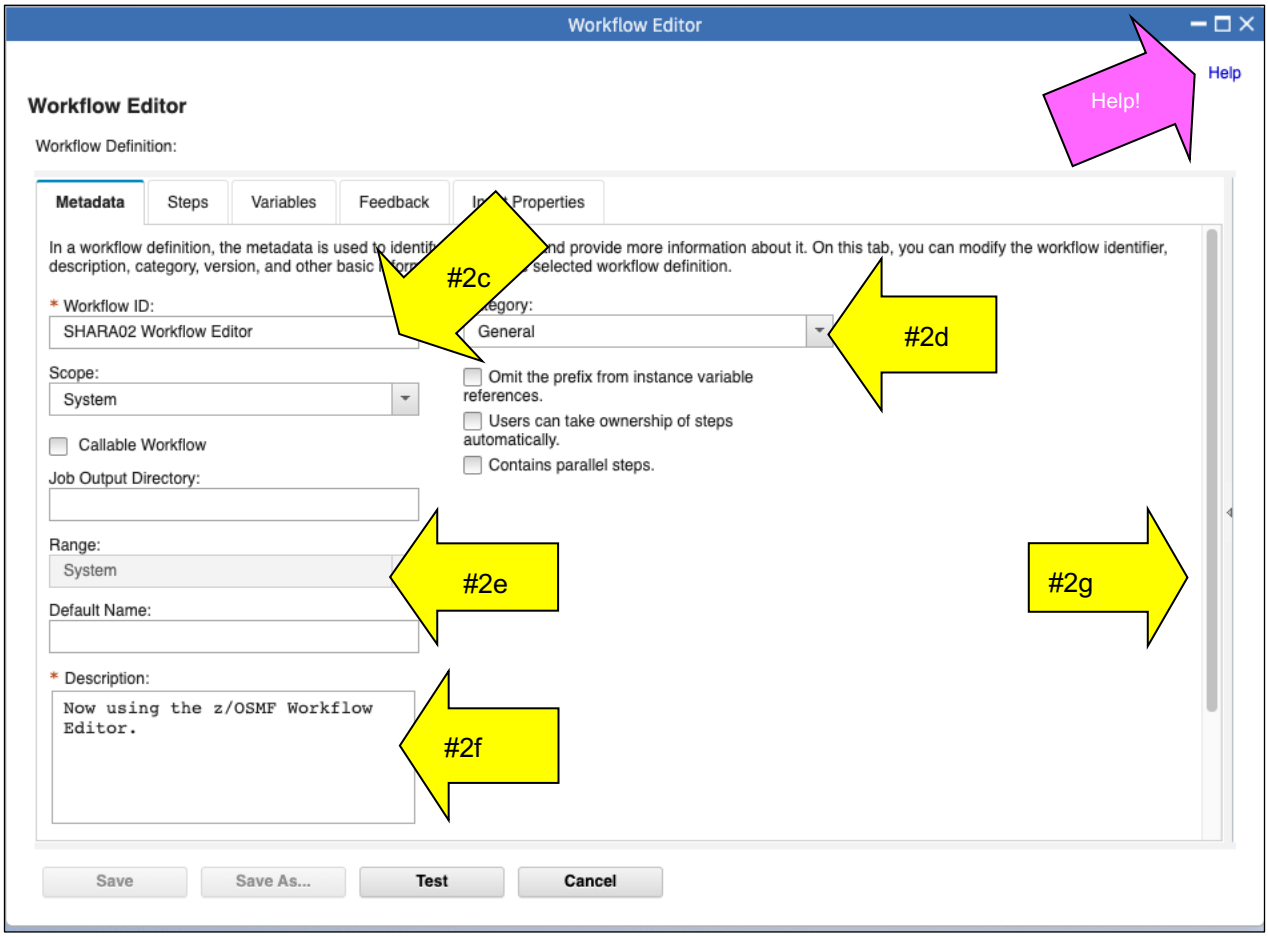
- A suggested name: ***YourUserID Workflow Editor Lab***, where ***YourUserID*** is your assigned userid, such as SHARA02.

d. Let’s make this Category **General.** A General Category just means it’s not identified as doing Configuration or Provisioning. General is fine for your first Workflow, and what you probably want to use for your own Workflows that perform normal tasks on your systems.

e. Scope will be **System** here. System means that a maximum of one instance of this workflow can exist on any one system in the sysplex.

f. Type in something you like for **Description**. If you type too much, the Editor will tell you it’s invalid.

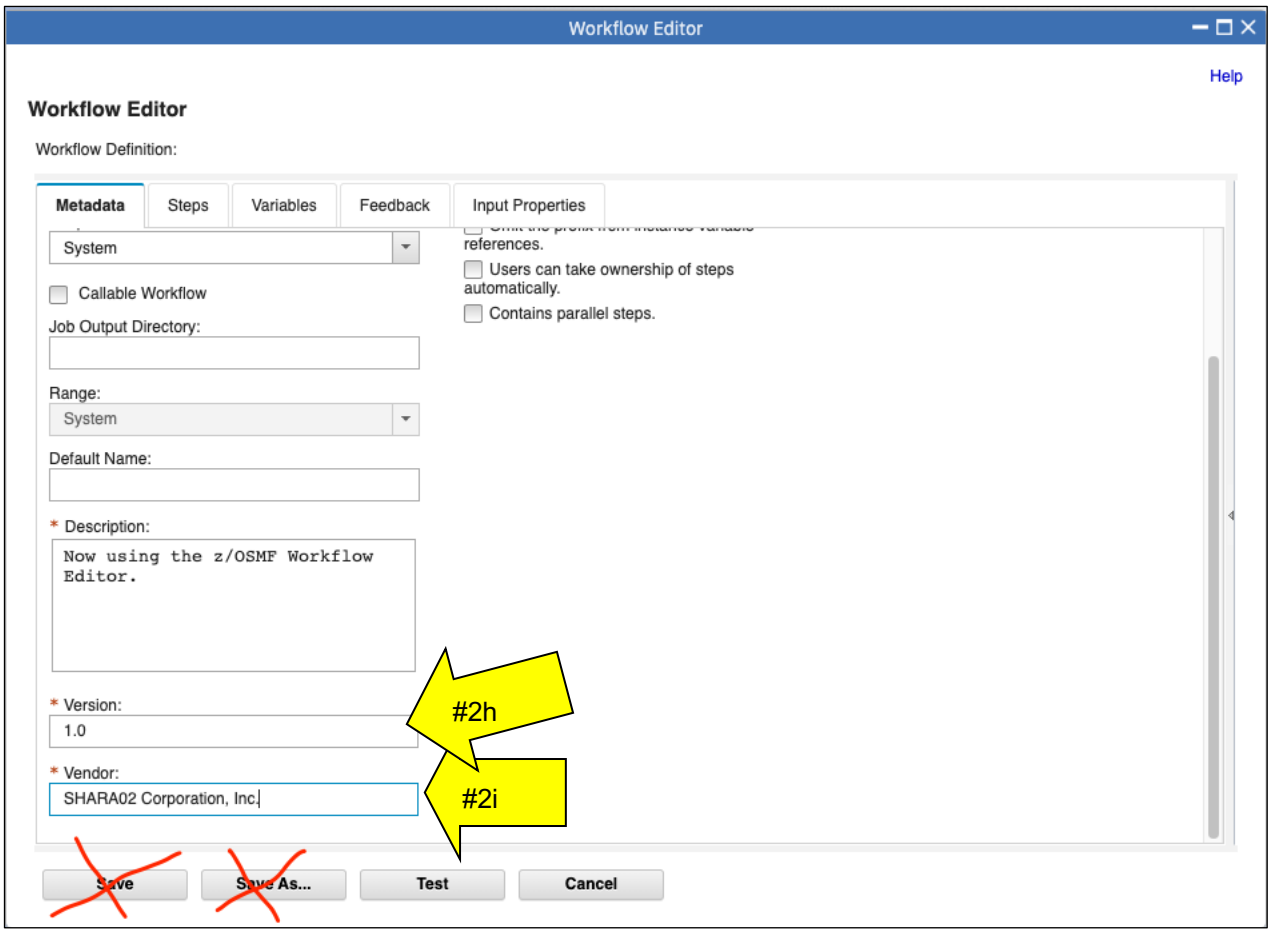
g. Scroll down to see more information.



h. Type in something you like for **Version**, such as 1.0. Think of this as the SMP/E REWORK value, if you like, and you are familiar with SMP/E.

i. Type in something you like for **Vendor.** Who designed this Workflow? What company created this Workflow?

Do not click on "Save" at this time. This lab will take you through each tab.



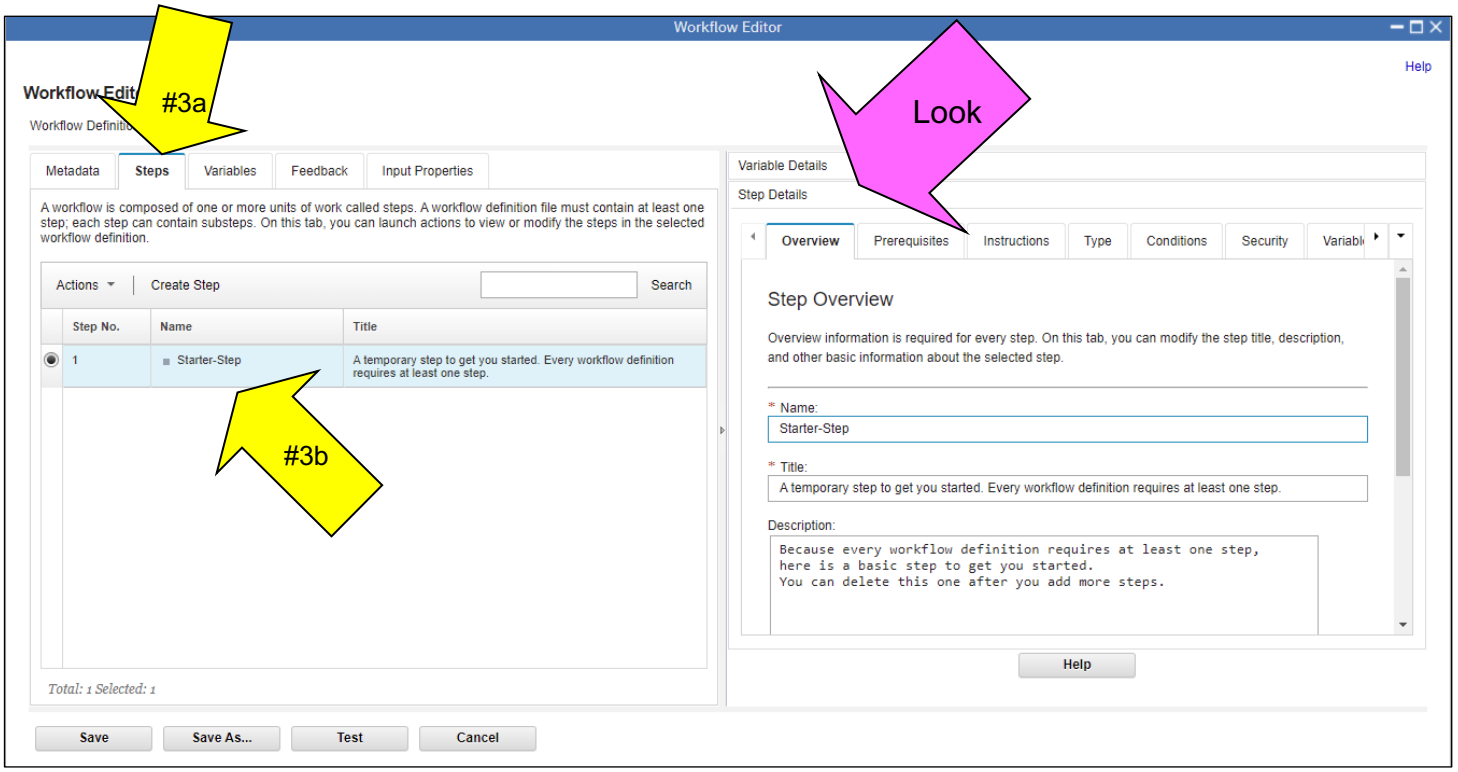
3. Adding Steps to your Workflow.

Now, we'll add some simple steps to our new Workflow.

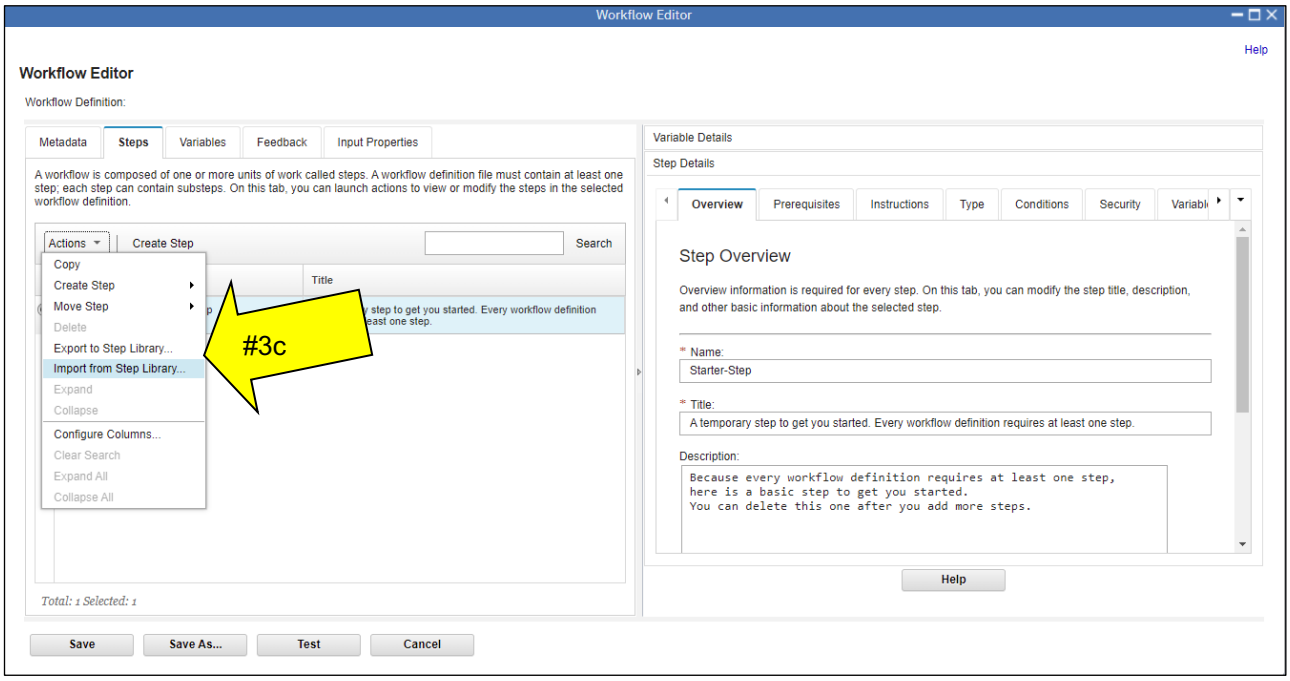
a. Click on the **Steps** tab. It might not be obvious now, but these tabs play a large role in creating your Workflow. You need to go through those tabs.

b. You'll see that you already have a "**Starter-Step"** in this empty Workflow, given to you by default. That is because you have to have at least one step in a Workflow. Click on the "**Starter-Step"** and notice that "**Step Details"** appear at the right.

- Notice that there are two detail "tabs" on the right: **Step Details** and **Variable Details.** Those two tabs on the right are very helpful to tell you what that step on the left is doing and what it concerns.



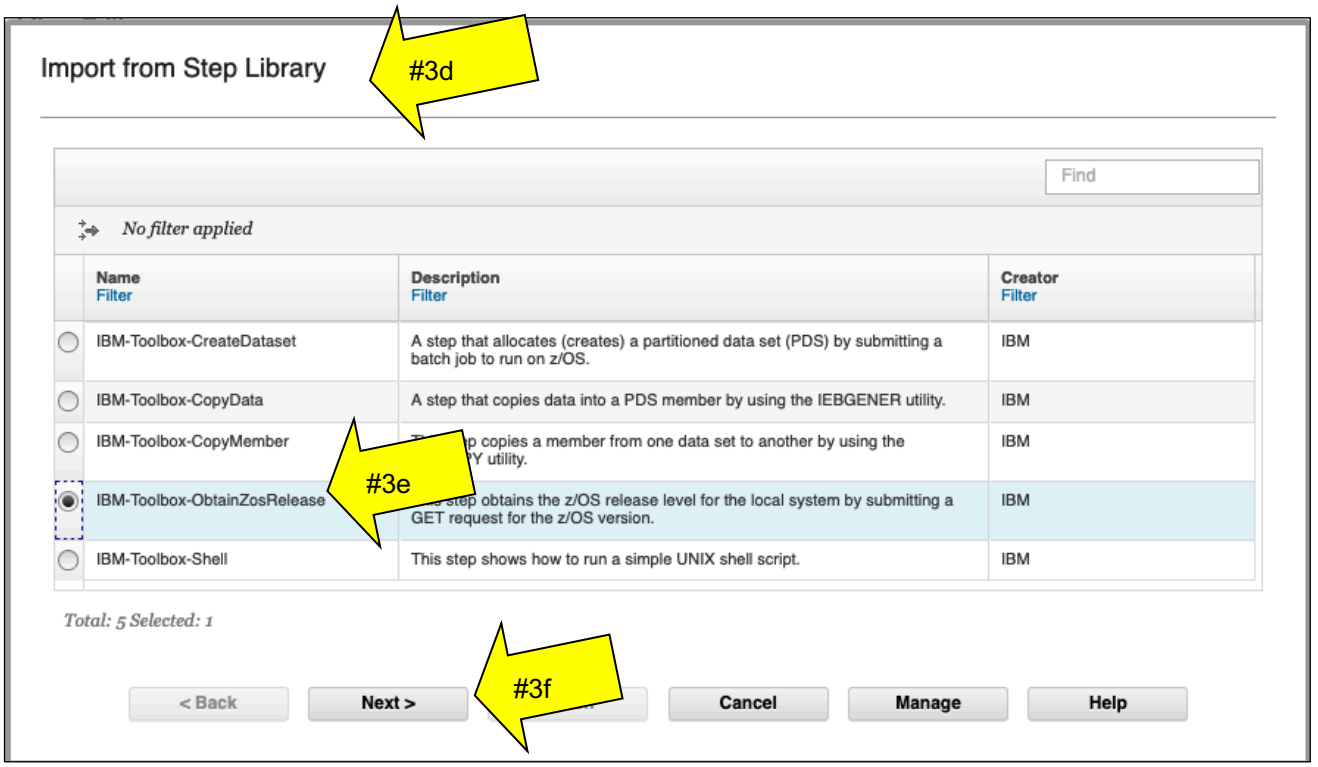
c. Click on **Actions->Import from Step Library…** to add the first “real” step to the new Workflow.



d. A Step Library selection box will now appear with various tasks you can choose from.

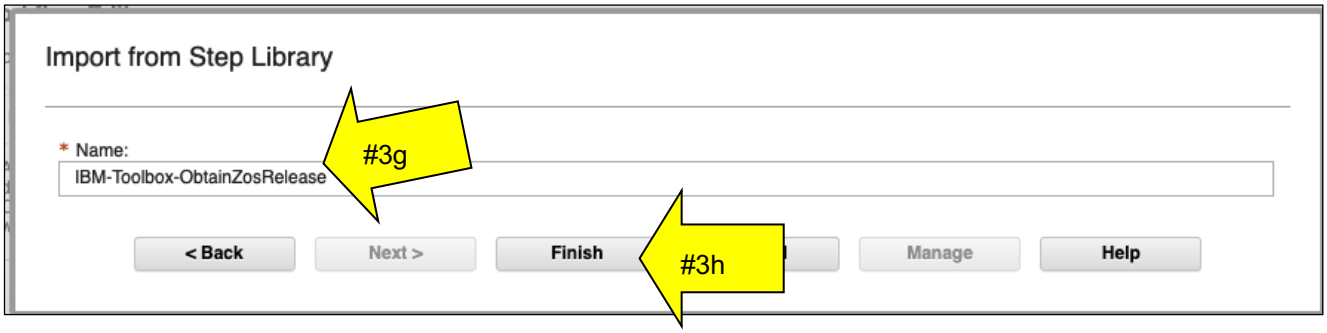
e. For this lab, Select IBM-Toolbox-ObtainZosRelease.

f. Click Next

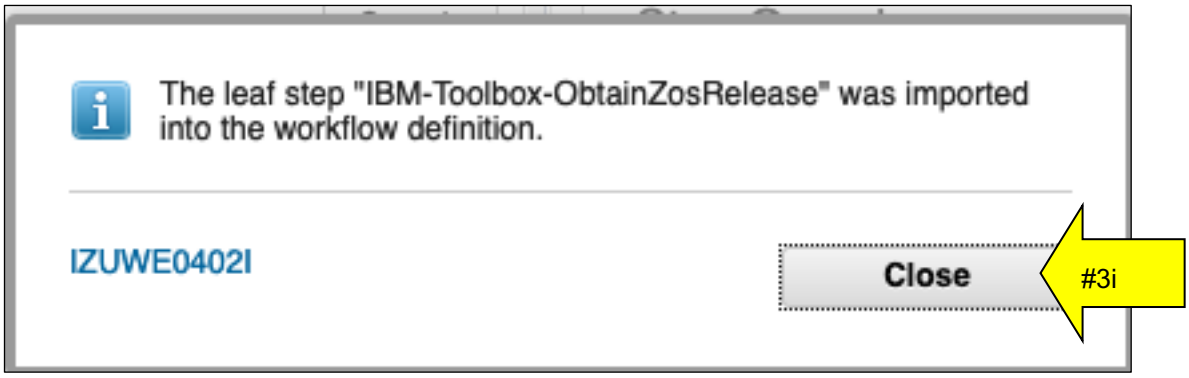


g. Give your step a name, or you can use the default that is there.

h. Click Finish

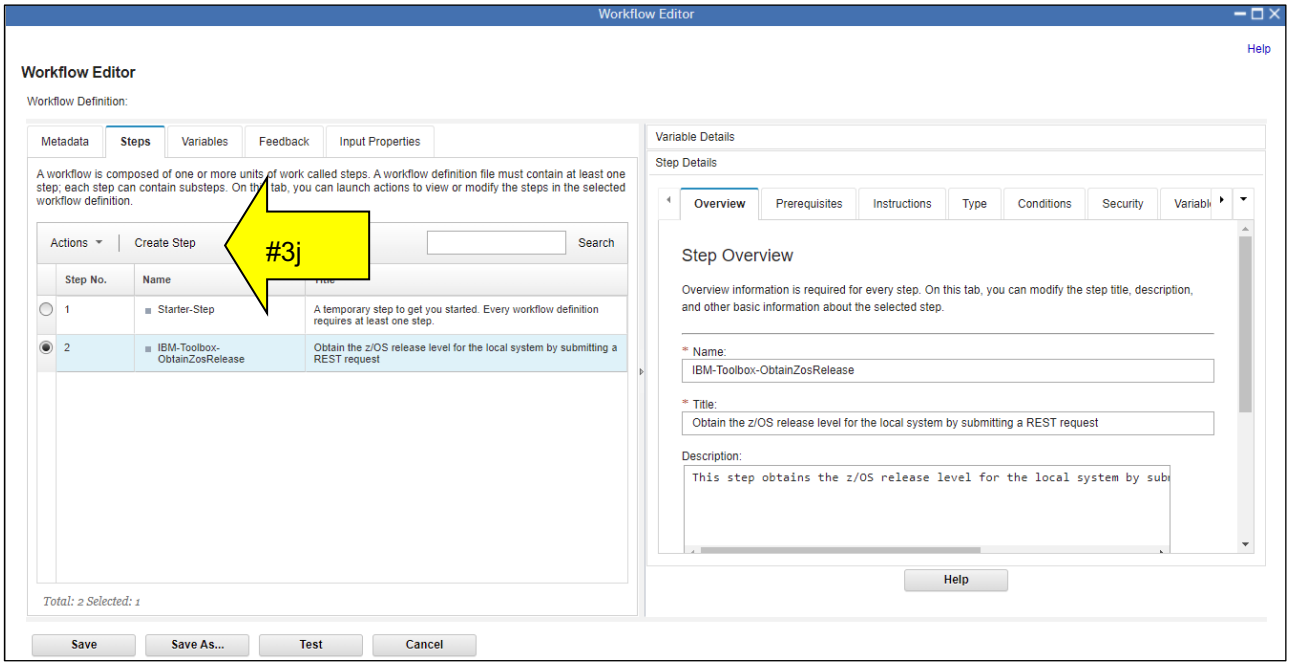


i. You will then see this. Click **Close.**



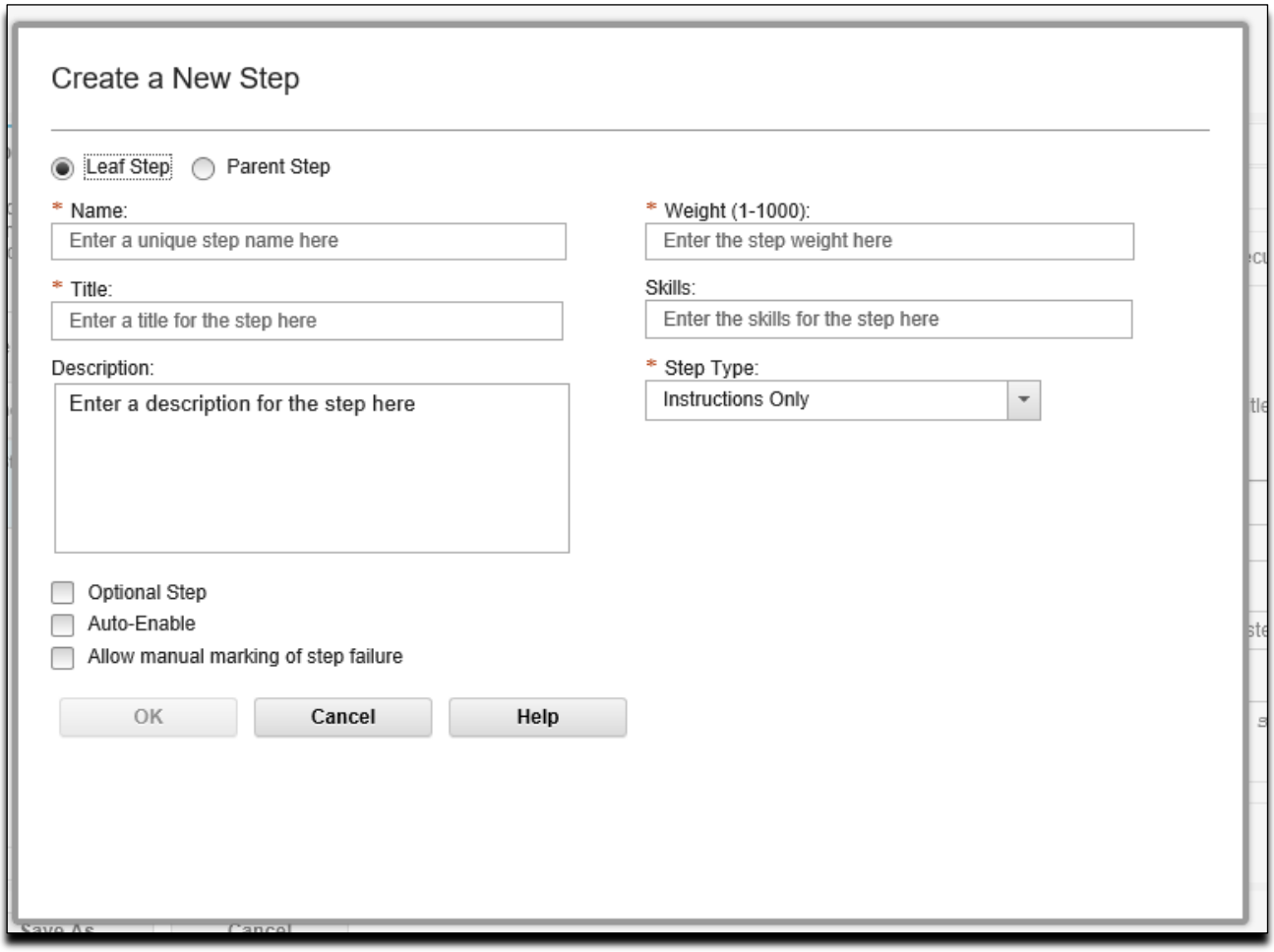
Now, let's add another step.

j. Click on **Create Step** to add another step to the new Workflow.



k. This step dialog box now appears. You have to decide if you want a **Leaf Step** or a **Parent Step. Leaf Step** is selected by default. Select **Parent Step** radio button, so we can try out a Parent Step creation. (The Leaf Step button is the default, so it will be pre-clicked.)

- A **Leaf Step** is simply a step that a user will actual perform.
- A **Parent Step** is simply a grouping of steps (**Leaf Steps)** that you want to put together. You don't perform a **Parent Step**; you perform a **Leaf Step.**



l. On the **Parent Step** dialog, give your step a **Name**. Of course, there is flexibility for what you can name it, but so that your Workflow matches the Workflow in this lab, call it ***RunStuff***

- The name for a Workflow step must be unique and cannot contain spaces. The user will not see this step name, but it is needed by z/OSMF to know what this step is called in case you wanted to reference it somewhere else.

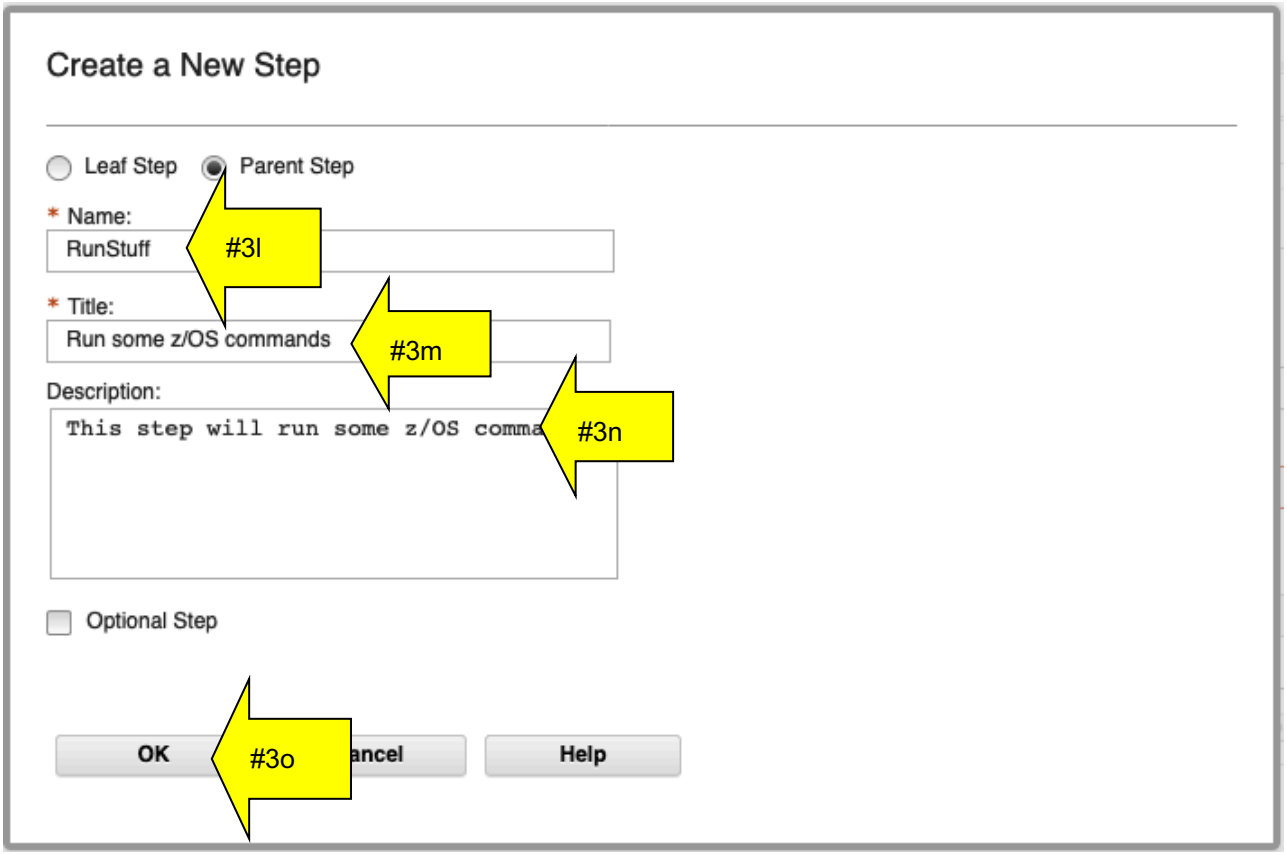
m. Provide a **Title.** Again, for simplicity and for matching these lab instructions call it ***Run some z/OS commands***

- Notice that this title can contain spaces. This is what the user will see as the step name on the Workflow. It can be up to 100 characters.

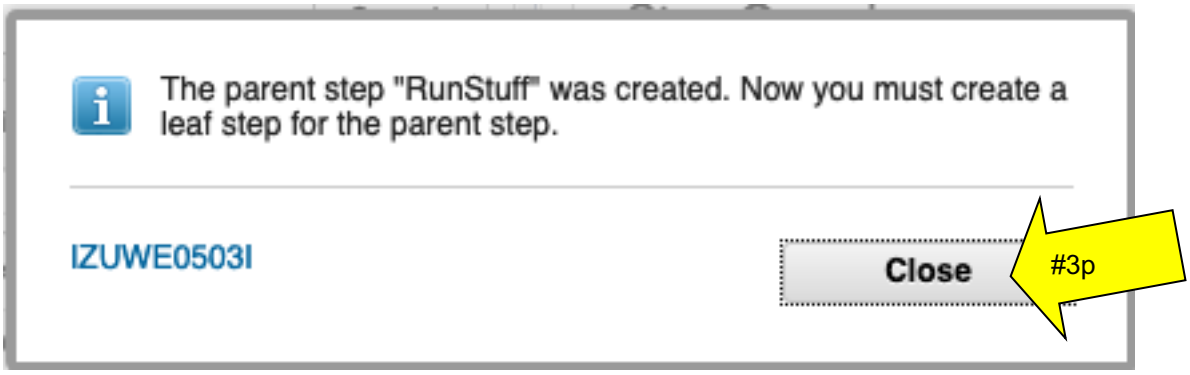
n. Provide a **Description.** Type whatever you want here.

- You can have a long description that is up to 500 characters.

o. Click on **OK.** This will add a parent step to your new Workflow under which you can now add Leaf Steps.



p. You will then see this. Click **Close.**



You can see (almost) in the background that your new parent step was created, however, the Workflow Editor now wants you to create a leaf step under the parent step. It presents the **Leaf Step** dialog to you.

q. Keep the **Leaf Step** radio button selected this time.

r. Give the leaf step a **Name**. Of course, there is flexibility for what you can name it, but so that your Workflow matches the Workflow in this lab, call it ***RunLISTCAT.***

- Same as for the parent step name, the leaf step name must be unique and cannot contain spaces. The user will not see this step name, but it is needed by z/OSMF to know what this step is called in case you wanted to reference it somewhere else.

s. Provide a **Title.** Again, for simplicity and for matching these lab instructions call it ***Run LISTCAT against a z/OS data set.***

- Notice that this title can contain spaces. This is what the user will see as the brief step name on the Workflow. It can be up to 100 characters.

t. Provide a **Description.** Type whatever you want here.

- You can have a long description that is up to 500 characters.

u. Provide a **Weight.** Type whatever you want here. Allocating a z/OS data set is easy, so its weight probably isn't a lot.

- This is how difficult you think that this step is from 1 – 1000. Usually smaller the weight, the easier the step is to do.

v. Provide a **Skills** description. Type whatever you want here.

- You can put whatever you think is the right kind of skills it takes to do this step.

w. Indicate a **Step Type** of **File Template.**

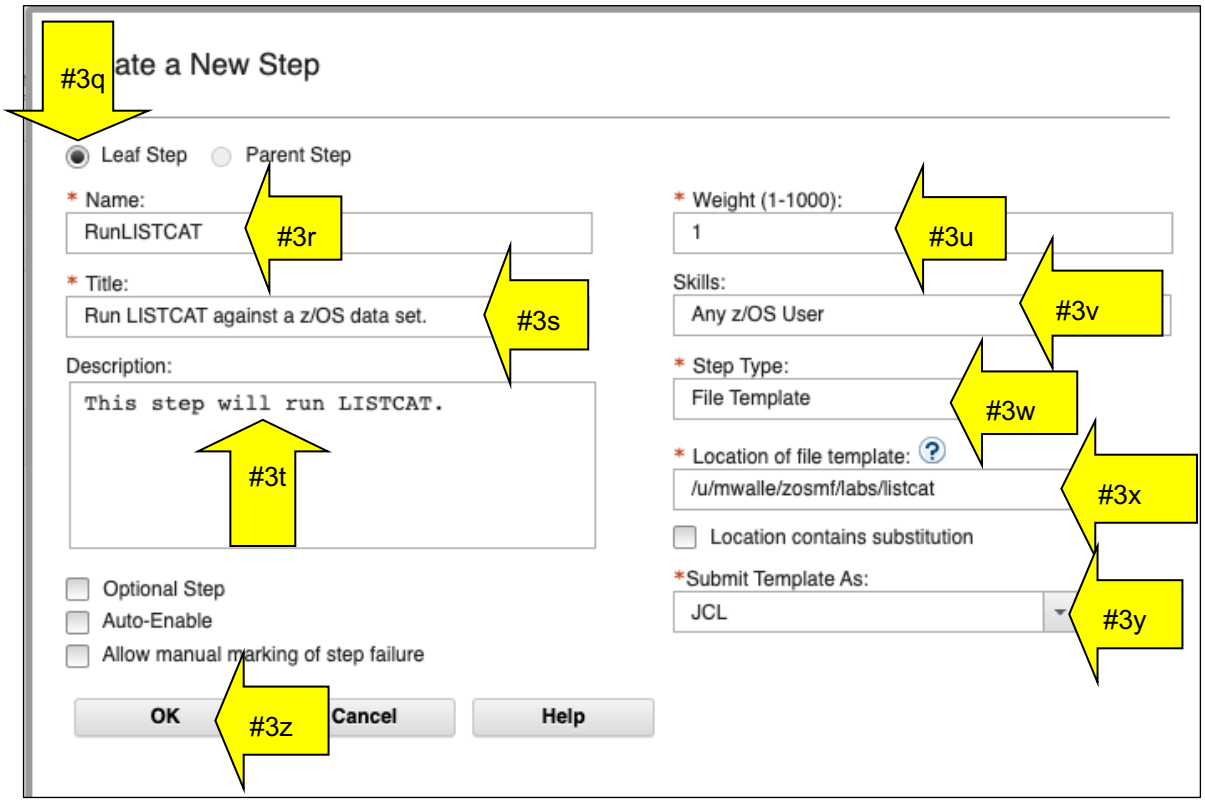
- A **File Template** will be the JCL that we provide (with some variables in it) from a file already created for this purpose. Traditional JCL would fit into the **File Template** category.
- There are other **Step Types** that you can explore later. For your very first Workflow, knowing how to submit JCL as a **File Template** can provide a lot of initial value, and is very easy.

x. When you have a **File Template,** you need to say where it is.

- Type in ***/u/mwalle/zosmf/labs/listcat*** if are doing this lab as part of a conference. If you are doing this lab independently you will need to add a file to your z/OS UNIX file system, which you can type in before completing this leaf step (so this step can find the file). The complete file for this step is shown in the Appendix of this lab.

y. Select **JCL** for **Submit Template As:**

z. Click on **OK.** This will add a leaf step under the parent step you created.



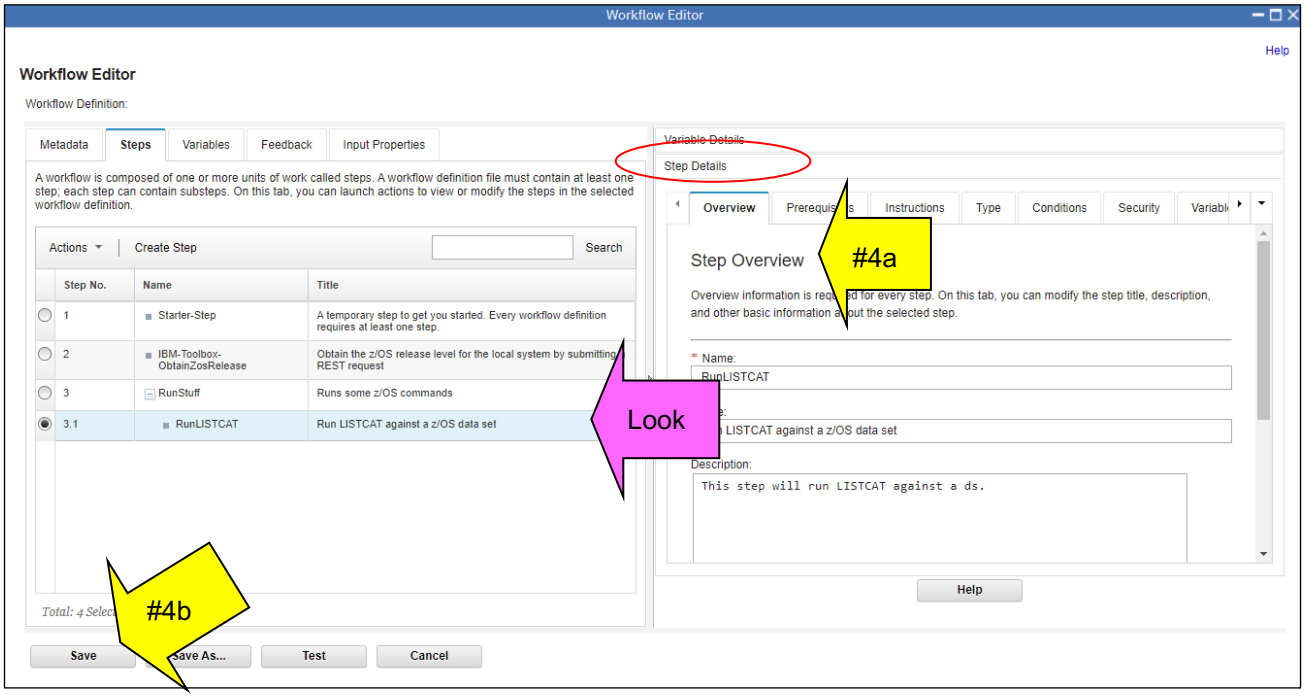
You will then see this. Click **Close.**

A wonderful thing to know! The Workflow Editor will never allow you to produce an invalid Workflow. In other words, if you use the Editor you can be assured that a user can create a Workflow that you produced. It might not be what you had intended to create, but that is a different story ☺.

4. Adding more step information to your Workflow.

You've got a parent step with one leaf step under it, and you can see that in Workflow Editor! Let's look at how to add a little more information to that one leaf step at this point.

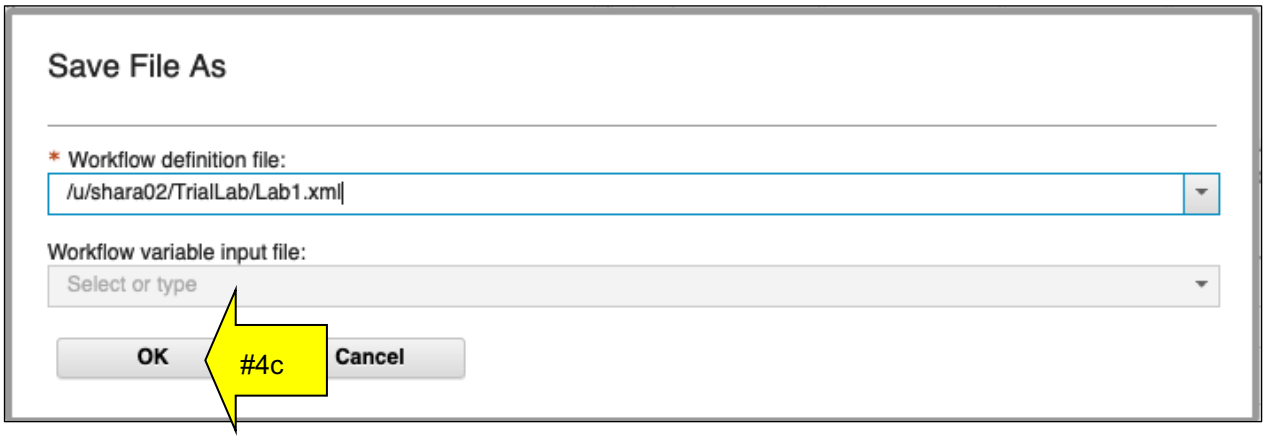
a. Under the **Step Details** and **Overview** tab on the right-hand side, browse through what is there. This is the information that you provided on the dialog box when you created the step. But there is a lot more you can do under **Step Details** as you will now find out. Just scan **Overview** for now.



b. First though, let's save our work! Click on **Save.** You should save throughout this lab, as you desire. If are able to click on the **Save** button that means you have something valid to save.

c. Type a location to save your Workflow definition file so far. You must have permission to write the directory, **and** previously that directory must be empty. (The Editor has added a subsequent enhancement later that didn't require an empty directory.) The Editor will create any directories, if they don't already exist. You can see the location where you have saved on the Editor primary panel (circled above). Then click on **OK.**

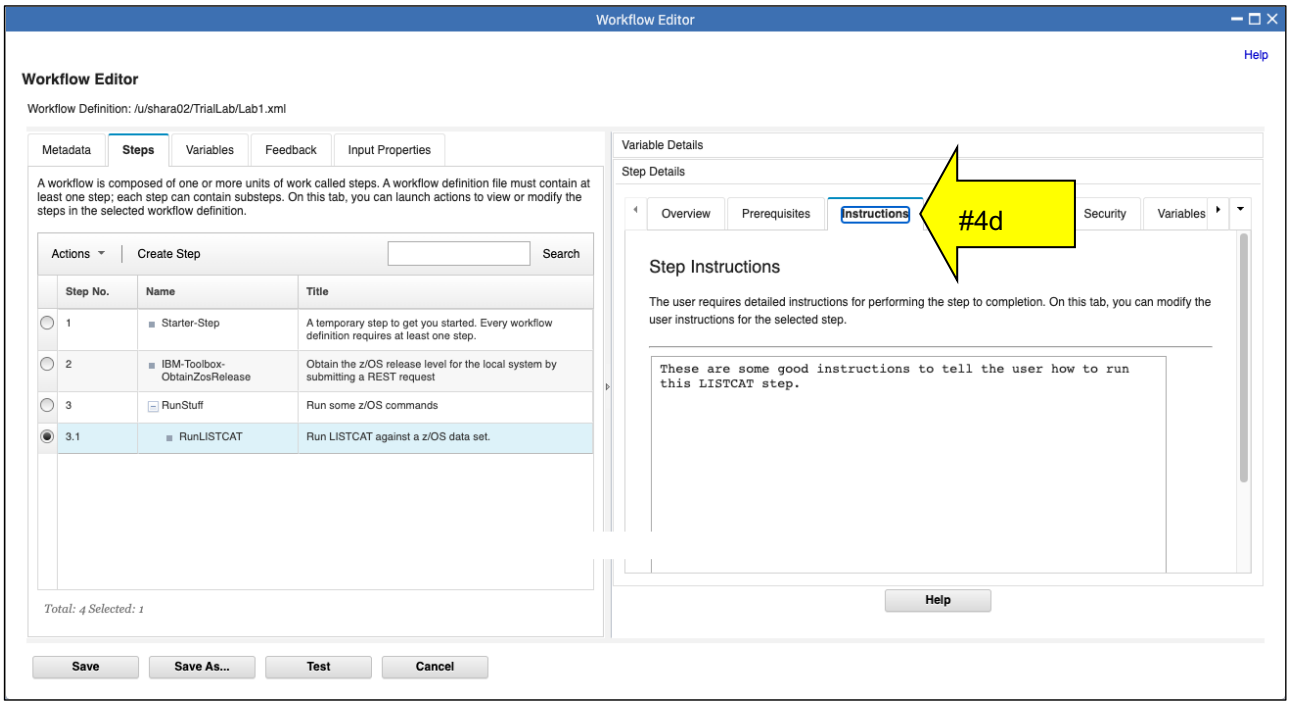
- If you are doing this lab as part of a conference, use ***/u/YourUserId/TrialLab/Lab1.xml*** as the location, where ***YourUserId*** is your assigned userid. Use all lower case for ***YourUserId*** , such as ***sharea01.*** Remember, pathnames are case sensitive.



Let's create some more **Step Details** for our ***RunLISTCAT*** step. For this portion of the lab, we will be on the right-hand side of the Editor.

d. Click on **Instructions.** Fill in some helpful documentation that you want the user to see when they are performing this step. Note that in the instructions provided below for our sample Workflow, we are telling the user what input they should be prepared to answer.

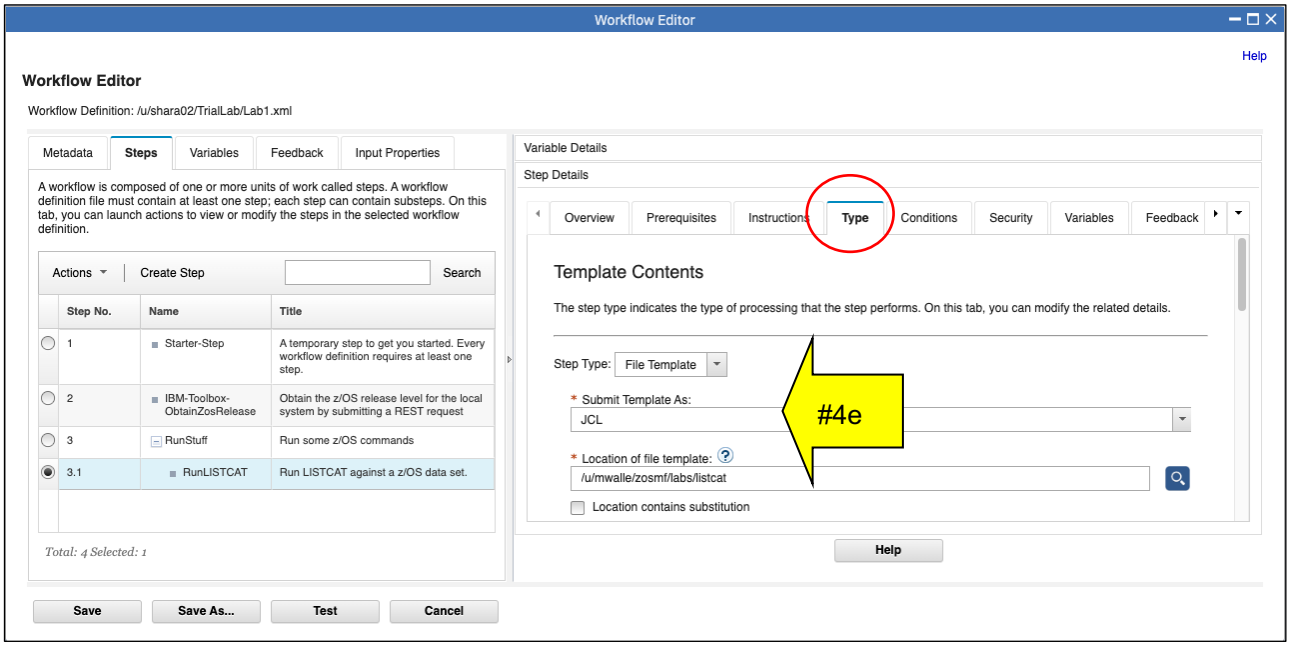
- *What are good instructions?* Here, a user will want to know how to perform this step. For instance, in our Workflow we are running LISTCAT, the user will have to be prepared to supply one variable (a data set name). This is the type of helpful information that good instructions should contain.



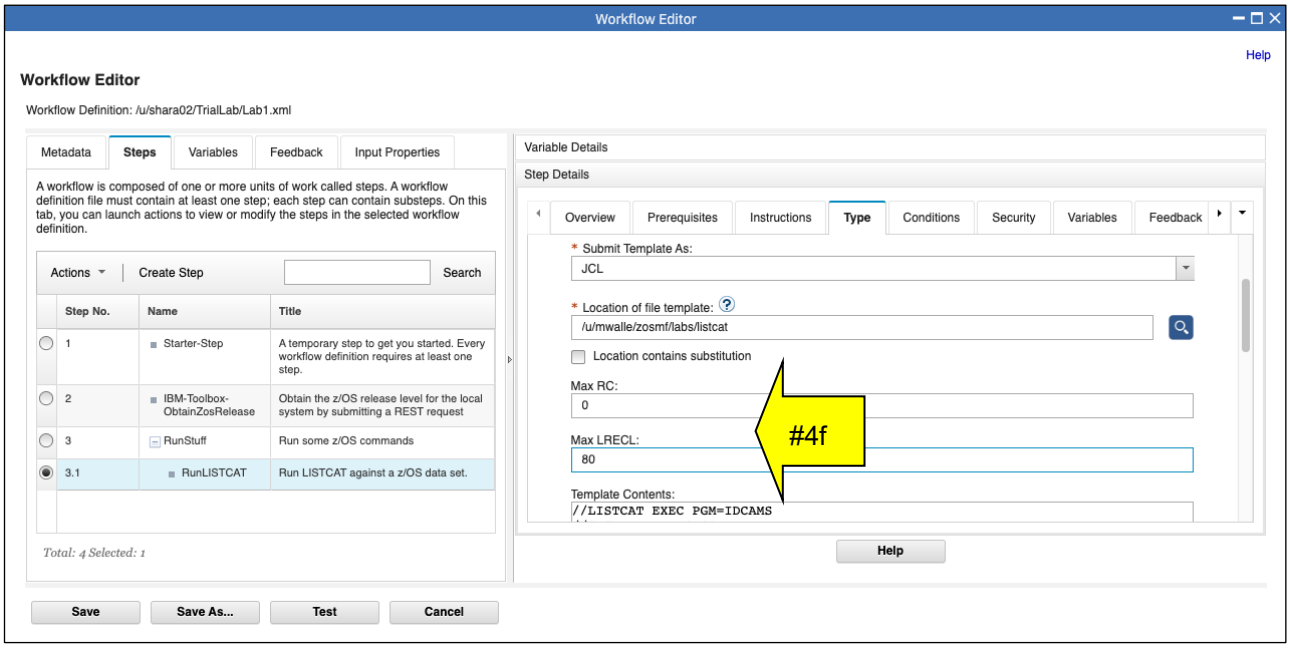
Next up under **Step Details** for our ***RunLISTCAT*** step is **Type.** This is an important tab for our sample Workflow because we need to specify some things that we were not able to during step creation.

e. Under **Submit Template As:** select the **JCL** drop down.

- This is saying that the template that we are using (which we said when the step was created, **Path Name of File Template**), is going to be **JCL** for the Workflow to submit. There are other kinds of "work" you do on this step, such as REXX or shell command.

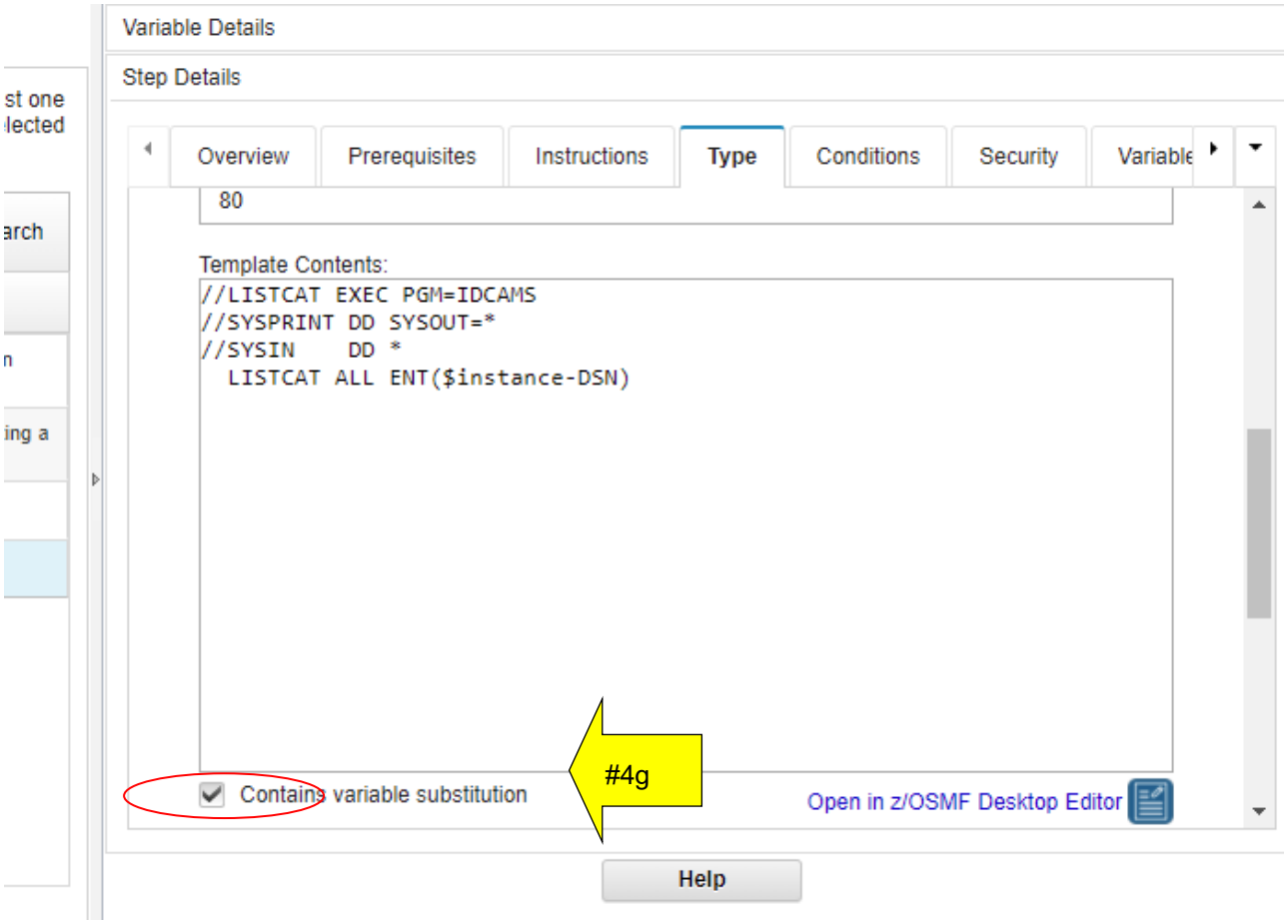


f. Scroll down. Under **Max RC**. We want this to be 0, since this step must succeed for the rest of our Workflow steps to work. Put a **0** in that **MAX RC** box. Make **Max LRECL 80.**



g. Keep scrolling in **Step Details** to the **Template Contents.** This was pulled in from the **Path Name of File Template** location during step creation and you can see it here. Notice that there are variables that need to be replaced with "real values", For that reason, you must check **Contains variable substitution.** This is important, or the JCL would try to run as you see it and it would fail!

Note that if you don't see anything in the **Template Contents:** that means you didn't type the right template name in the **Path Name of File Template** location. You should see the information below.



Nothing else on this tab needs to be changed. We don't need to do anything in the **Conditions** or **Security** tabs.

5. Variables in your Workflow.

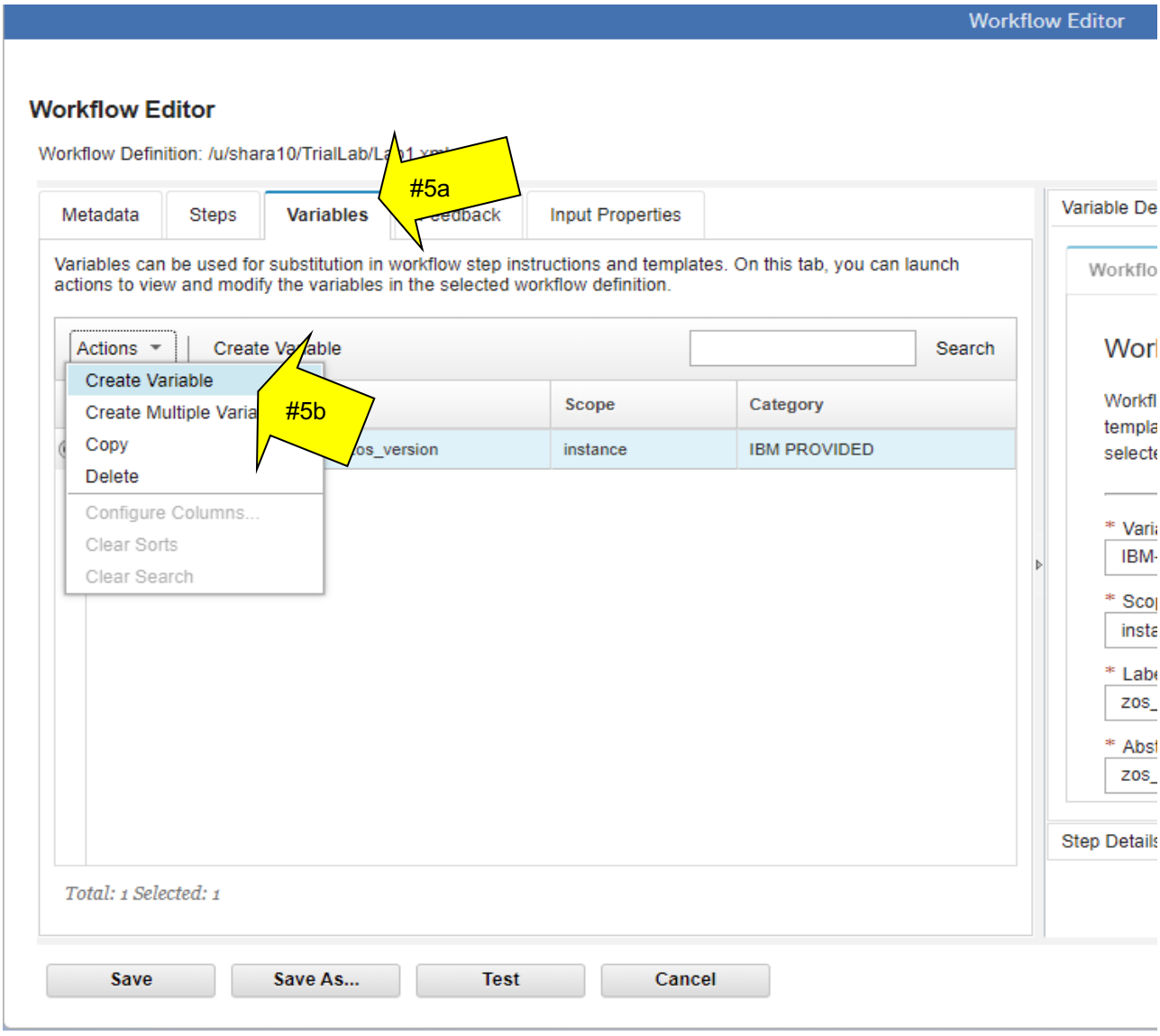
We have variables in our Workflow steps, if you looked closely in the JCL template. That is how you bring value to Workflow steps and allow the user to give you some input. You put that input into a variable and then substitute it in the template when it is time to run the JCL and is then submitted by the Workflow.

You can provide variables to a Workflow in two ways: you can individually create them here under **Variables** tab from the Workflow Editor, or you can pull them in from an external file (called "Input Properties) that has them already resolved. We'll not explore Input Properties in this lab.

Let's create variables in the Editor:

a. Click on the **Variables** tab.

b. Then, click on **Actions** drop down, and select **Create Variable.** Alternatively, you could also click on **Create Variable** right next to **Actions**.



c. The **Create a New Variable** dialog opens up. Here's where we will define two variables that we will use in our sample Workflow. Fill in the values you see here, so that the lab is consistent with the screen shots. Enter ***DSN*** for **Variable Name.** You cannot use blanks in your variable names.

d. For Scope, select **Instance.** This means that the scope of the variable will be inside this instance of the Workflow. The other option is **Global**, which means that the variable can be used outside this instance.

e. For **Label** type ***Data set to run LISTCAT against.*** This will be the short name that the Workflow user will see as a description of what to fill in here.

f. For **Abstract** type what the user will see from the Workflow, to know what this field will be used for.

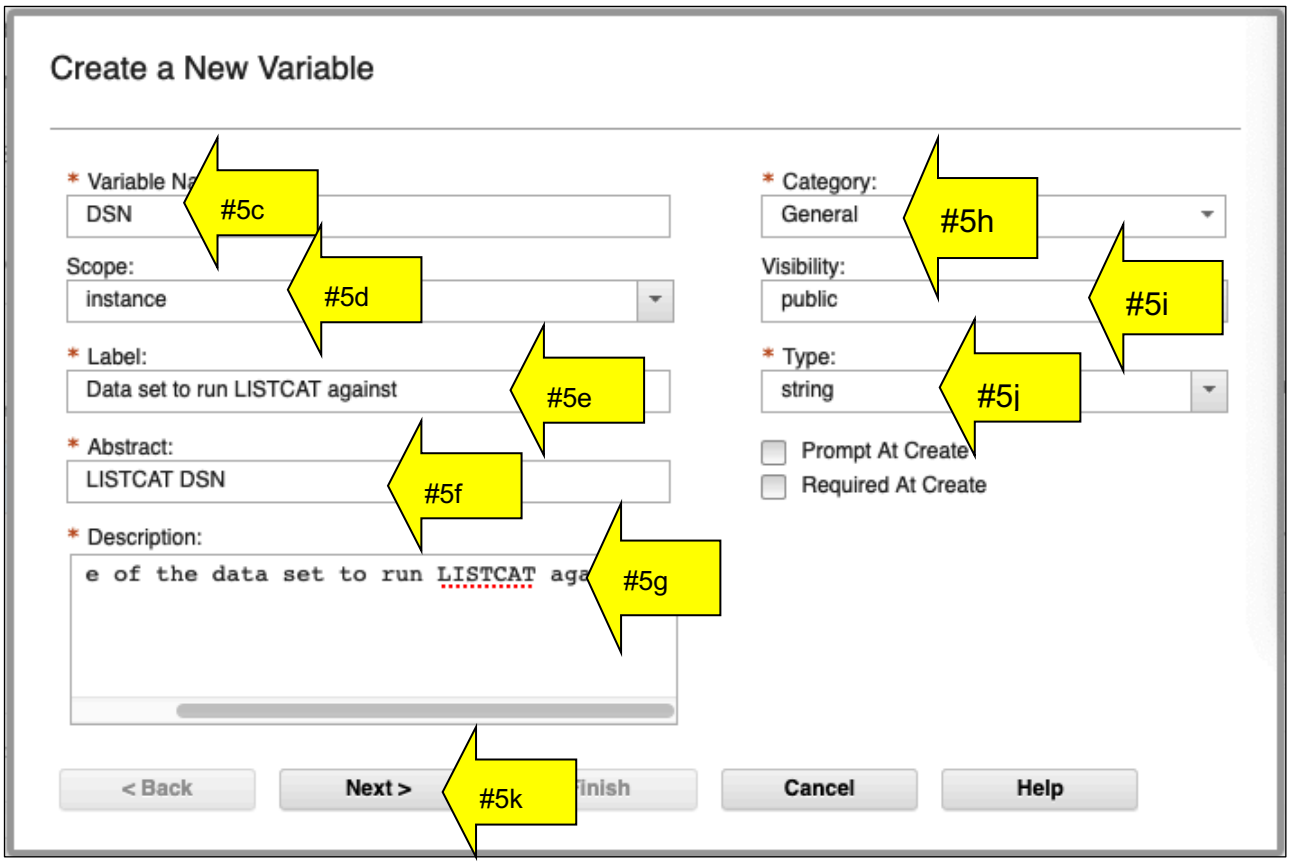
g. For **Description** type a longer description of what the user might need to know about this variable.

h. For **Category** select **General.**

i. For **Visibility** select **public.** This means that the user will see the variable while doing the Workflow. The other option is **private,** meaning that users won't see this variable in the Workflow.

j. For **Type** select **string.** The data set name is a string variable. There are other options for other kinds of variables. This will help the Workflow validate what the user must enter. Later, we'll make it more specific kind of string.

k. Click on **Next>.**

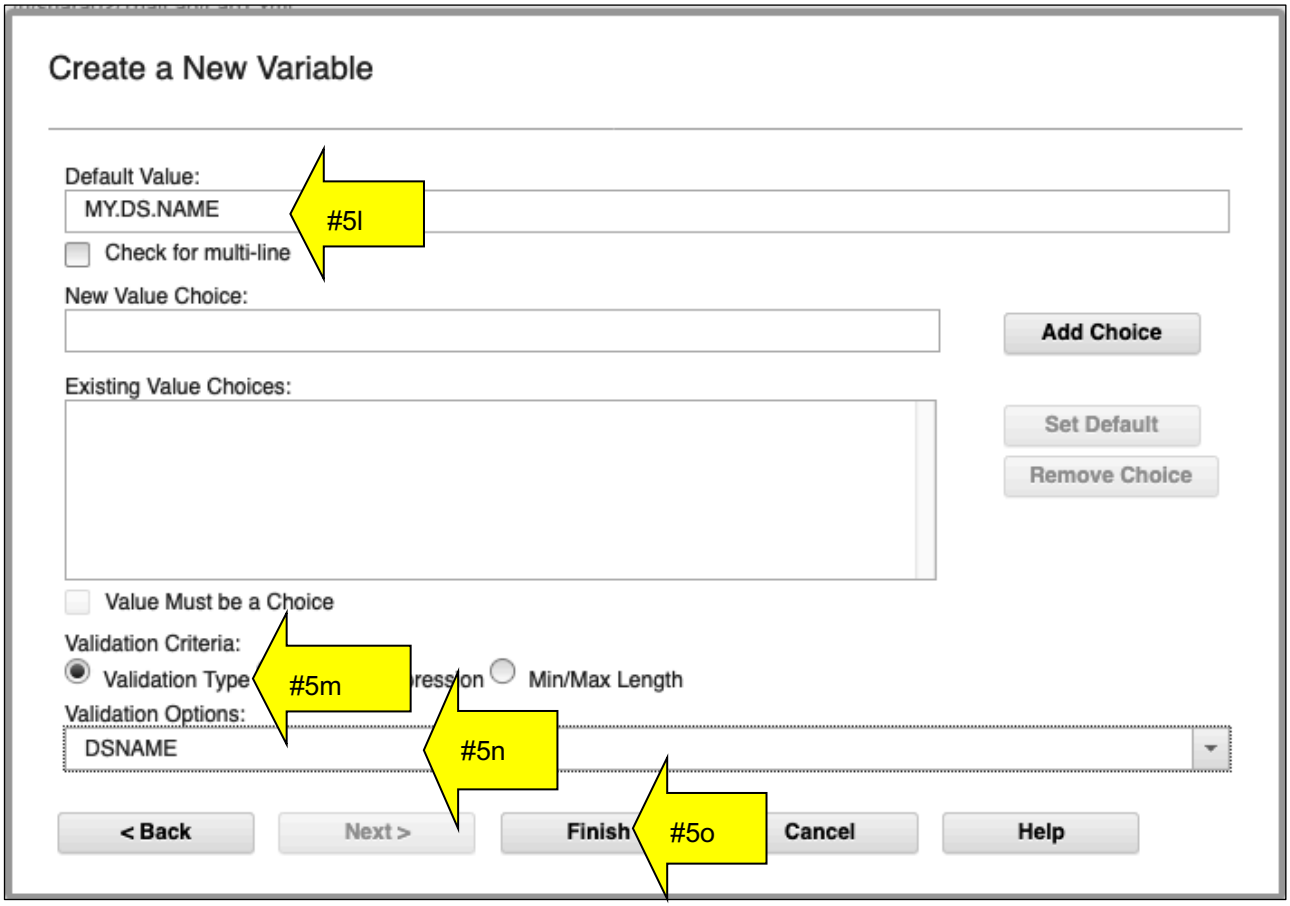


l. Continuing on in the dialog. Add a **Default Value** that you want for this data set, if the user doesn't supply one. Type **MY.DS.NAME** here, because that is the default name we want to use for this sample.

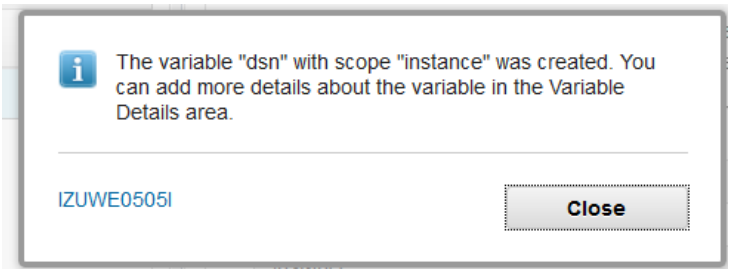
m. Under **Validation Criteria,** select **Validation Type.** This tells the Workflow that you want to validate the input from the user for this variable.

n. Then, you say that that the validation should be for a data set name. Under **Validation Option,** select **DSNAME.** This will make the Workflow make sure the user types in a valid z/OS data set name when overriding the default.

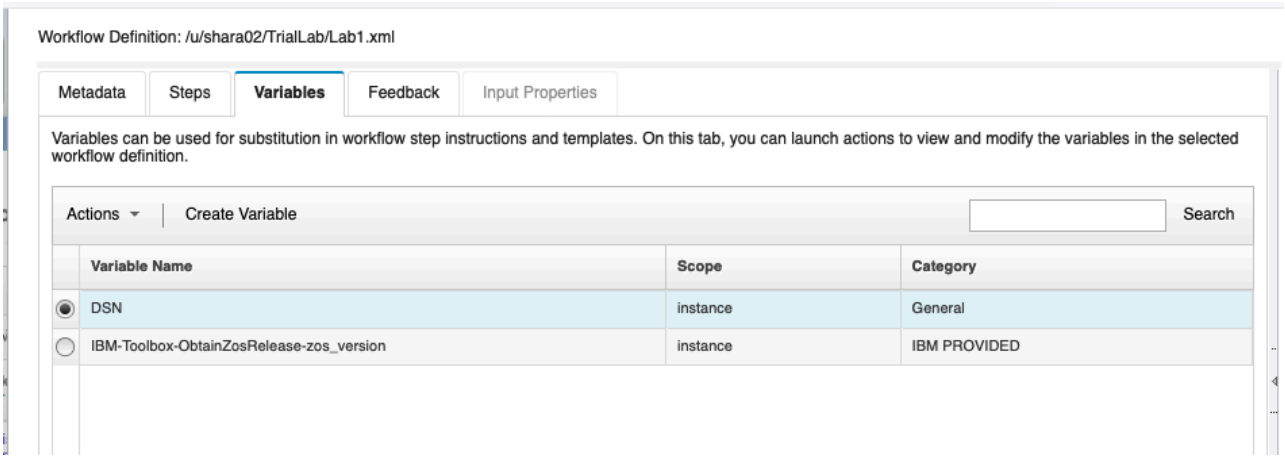
o. Finally, click on **Finish.**



You now created a Workflow variable. Click **Close**



When you are done, you should see two variables (***DSN and IBM-Toolbox-ObtainZosRelease-zos_version***) under the **Variables** tab.



An important thing to know about variables and Workflows:

Let's get back to that important check mark we used in our JCL template **Contains variable substitution.** Workflows do variable substitution, as you could have guessed, but how does it do it?

It uses a mechanism provided by the open-source Velocity engine created by the Apache Velocity Project. It can be pretty powerful and allow conditional directives, but we'll stick to the simplest form for our lab, which is straight substitution. How do you invoke the Velocity engine? By simply coding **$instance-*variable*** in your template where you want it. For instance, in our JCL template it is coded:

$instance-DSN

which the Workflow will substitute in the JCL to simply:

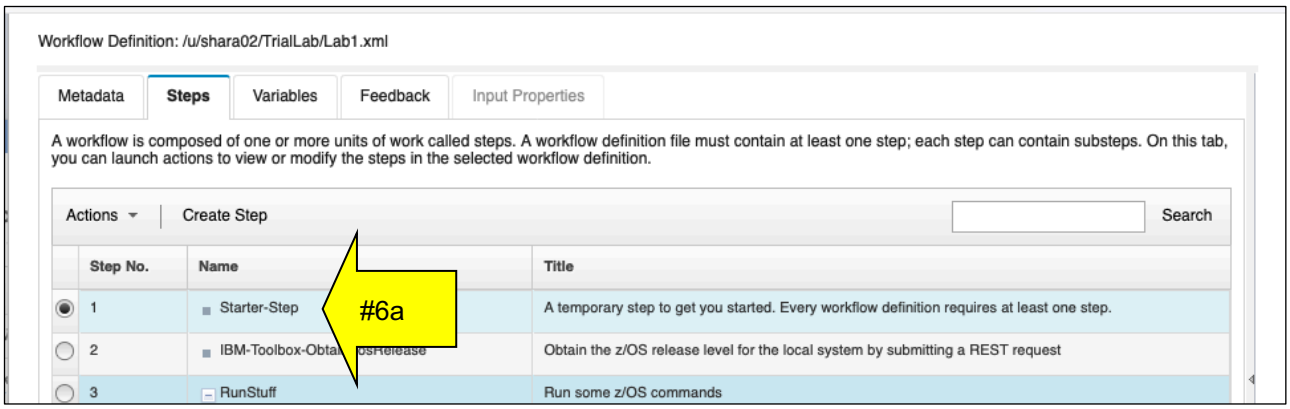
MY.DS.NAME (the default) or **MWALLE.LAB1.DSN** as appropriate.

6. Putting Variables in your Steps.

At this point, we have added one step (really one parent and one leaf step) and two variables in our Workflow. However, we don't have the variables *associated with any steps*. We need to put the variables in a step.

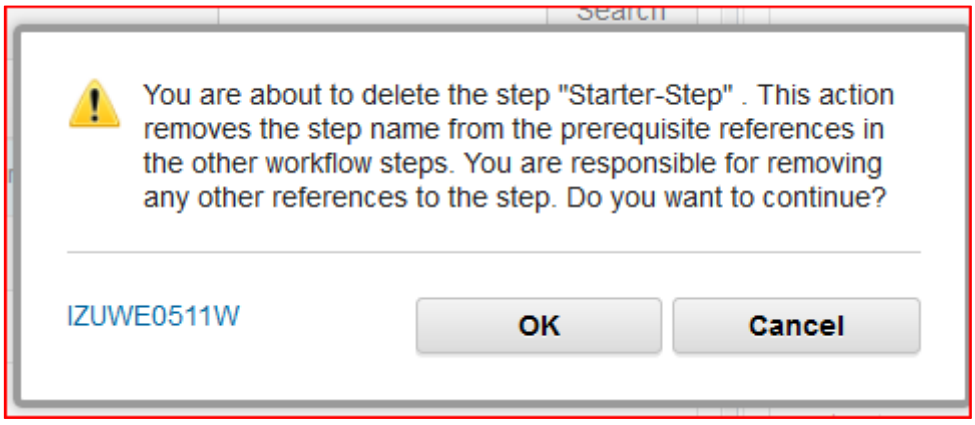
First, though, let's clean up our Workflow. Every new Workflow will have a starter step (called **Starter-Step)**, which we can see at the first step in the Workflow. Let's delete that starter step, because we don't need it. We have to have that step, as a Workflow *must* have a step if it is valid, and the Editor will only allow you to have a valid Workflow. Therefore, the Editor gives you a valid step, but it is expected that you'll delete it.

a. Click on the **Starter-Step,** which is step #1.



b. From **Actions**, select **Delete** from the drop down.

You will see the confirmation, click on **OK.** You now know how to delete a step in a Workflow you don't want.

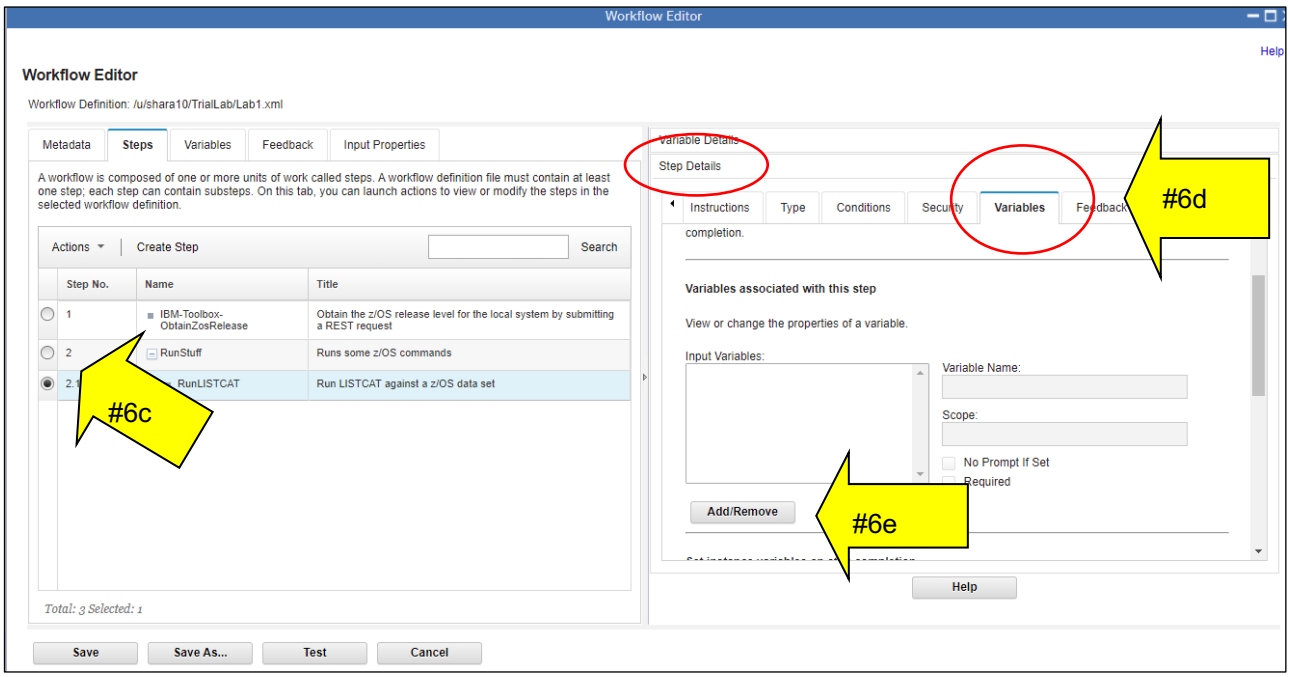


In your Workflow, you have one step you imported, and one parent step with one leaf step. Let's associate that leaf step with the variable that we have defined.

c. Click on the leaf step **RunLISTCAT**, which is now step #2.1. (You might have to untwist the parent step to see it.)

d. Click on the right-hand side's **Step Details,** and then **Variables.**

e. Scroll down to the bottom, so you can see the **Add/Remove** button. Click on **Add/Remove.**

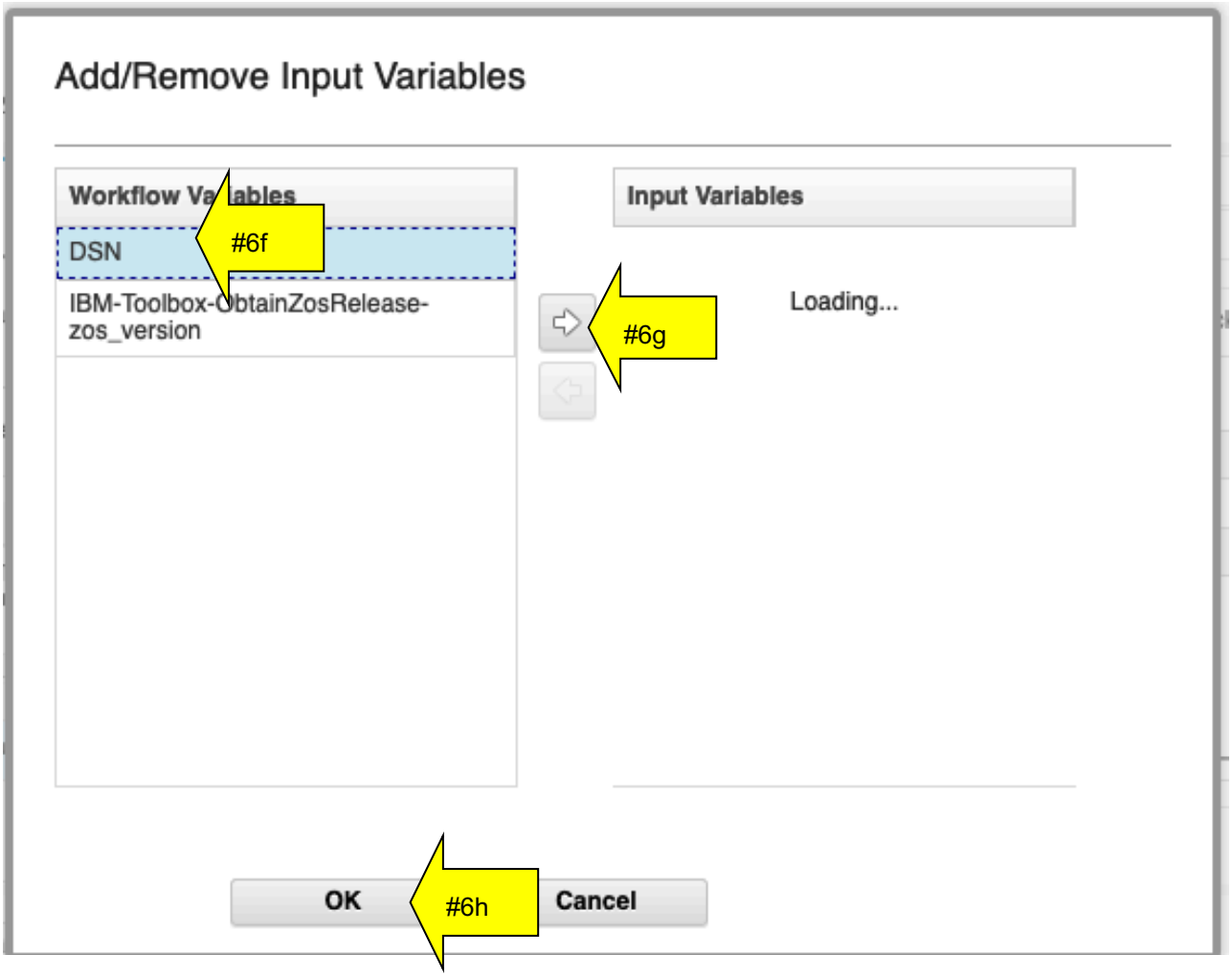


The **Add/Remove Input Variables** dialog appears. Look, there are the two variables, one added by the import step and one that we added. We need to move the one we added from the **Workflow Variables** column into the **Input Variables** column.

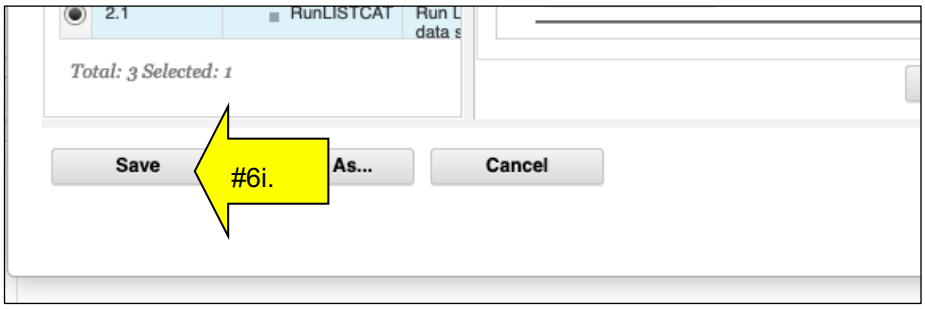
f. Click on ***DSN***.

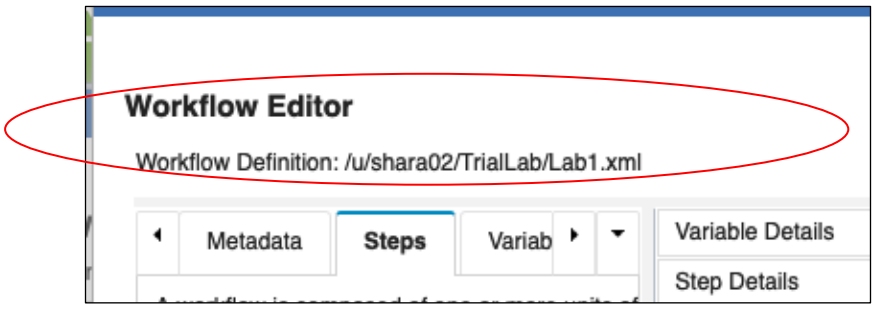
g. Then the **arrow** to move it to the other column.

h. Click on **OK** to complete the dialog.



i. Let's save the Workflow and see what we've created at this point. Back on the main Editor screen do a **Save.** The save will put your Workflow definition file into your existing directory as shown by **File Path:** (which was ***/u/YourUserId/TrialLab/Lab1.xml*** remembering to get the case correct.***)***



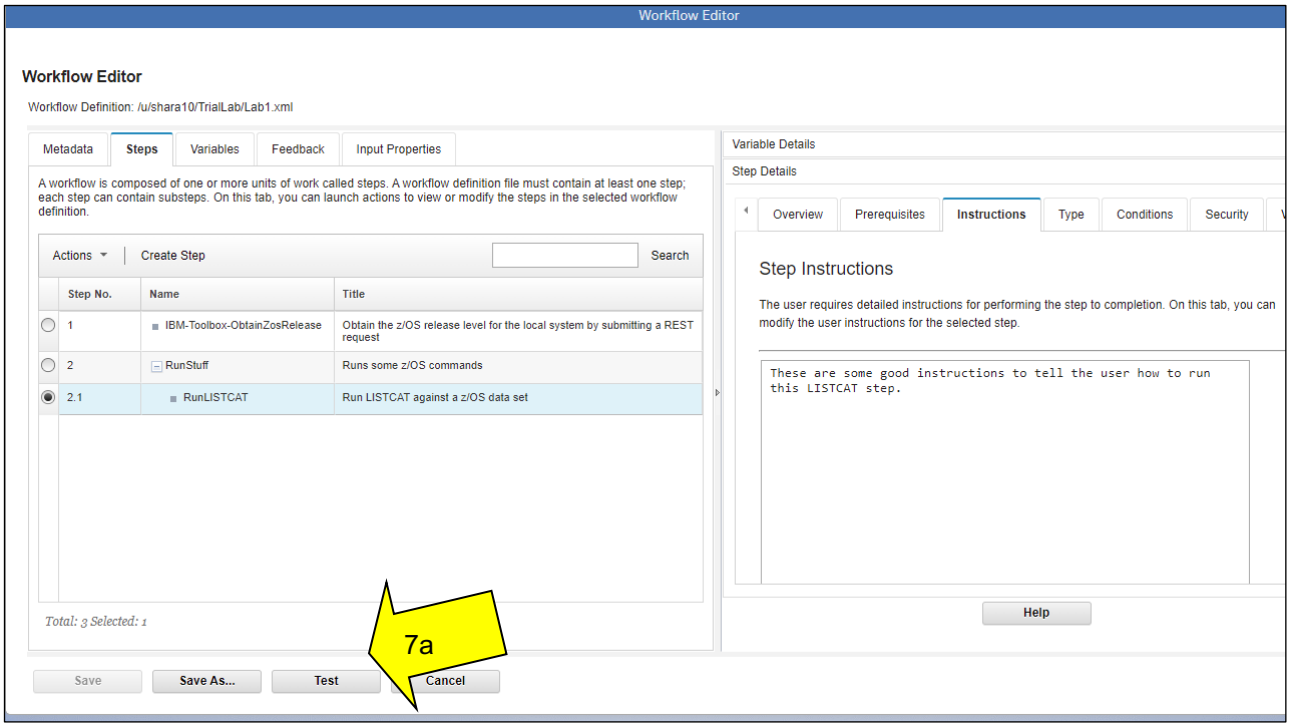


7. Trying out your Workflow.

You have a very basic Workflow completed. Let's try it out to see what it looks like to a user!

It is easy to try out your Workflow that you are working on with the **Test** button at the bottom.

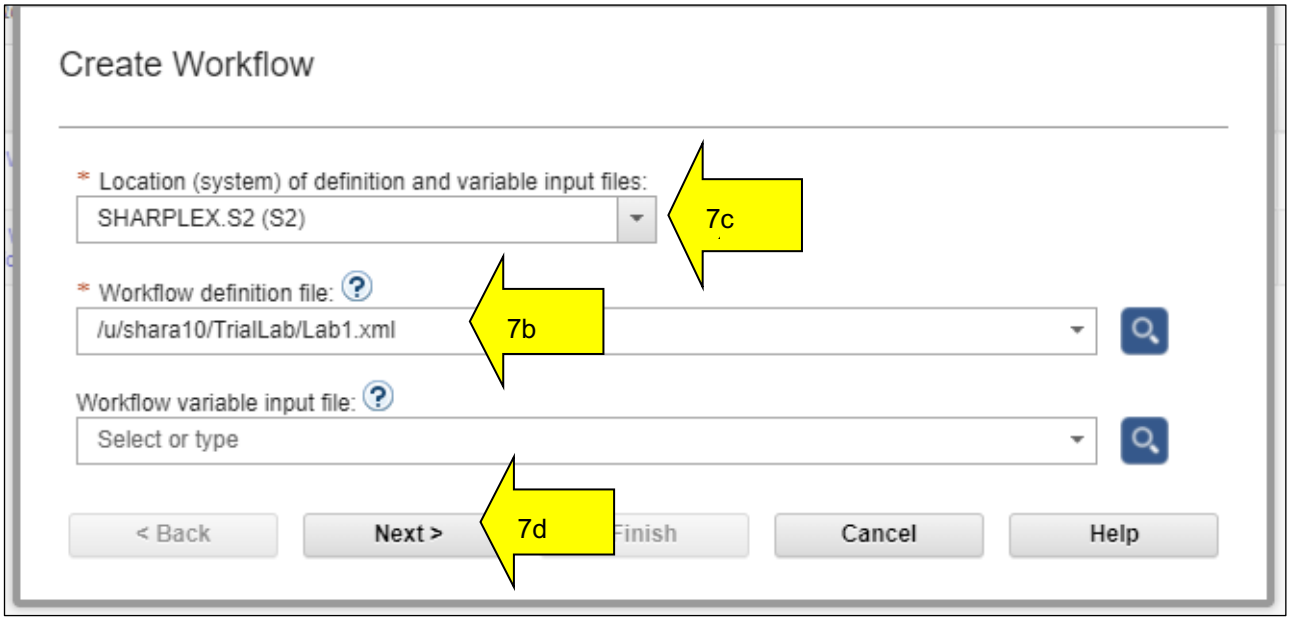
a. Click on **Test** to create a Workflow instance to see how your Workflow looks, which will launch you into Workflows.



b. On the **Create Workflow** dialog, verify the name of your Workflow definition file you just created. (which was ***/u/YourUserId/TrialLab/Lab1.xml)*** where ***YourUserId*** was the userid you were given for the lab.

c. Ensure the Location is **SHARPLEX.S2** for this lab system.

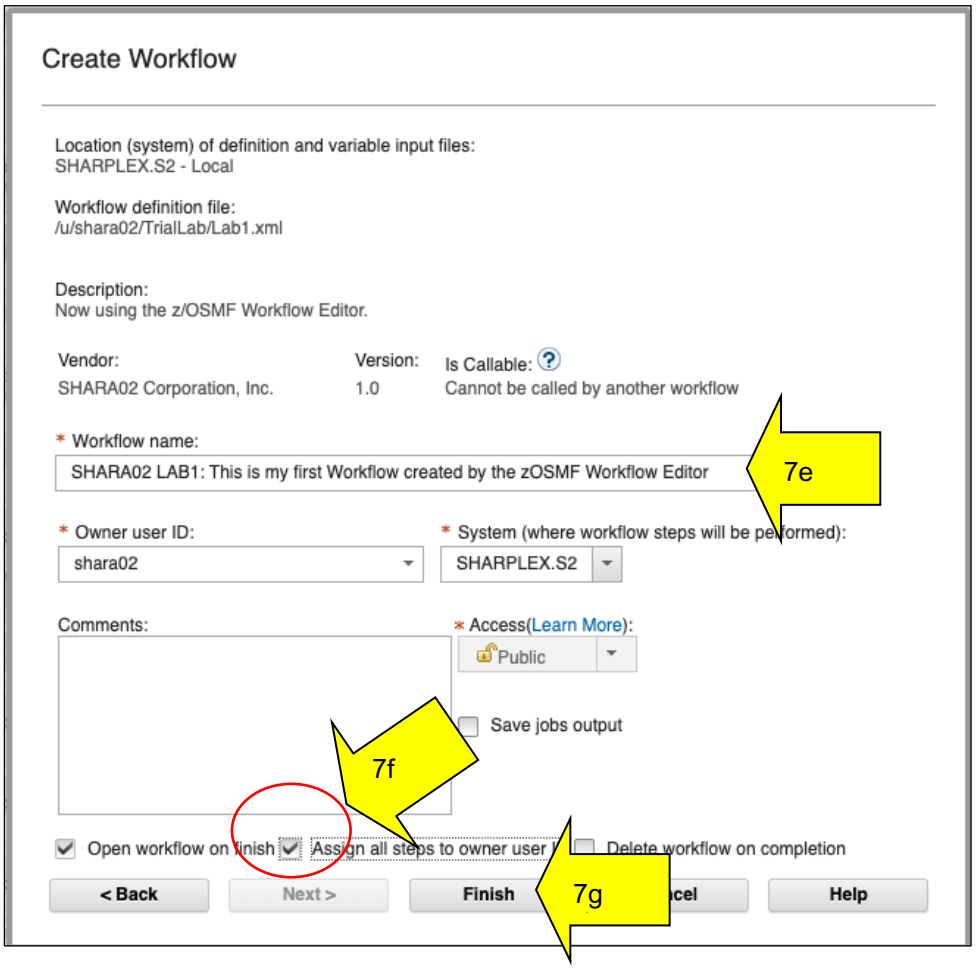
d. Click on **Next.**



e. Continue by filling in the **Workflow name:** Notice that this field is prefilled in with what you specified in the Editor, **Metadata** screen, as the **Description.** It may be that you see an error for this **Workflow name.** Currently, the Editor accepts characters for the **Description** that aren't valid for **Create Workflow** (like "/")**.** This is a known situation and easy to get past. Just remove the / and type a unique name for your new Workflow during creation (probably just removing the "/"). For instance, use: **YourUserid LAB1: This is my first Workflow created by the zOSMF Workflow Editor.** Notice that **–Workflow_0** is added by the Workflow creation process, and you did not have in your Editor **Metadata Description.** Be aware, you'll need to change the default name and Workflow ID each time you create a new Workflow definition (or you'll see an error).

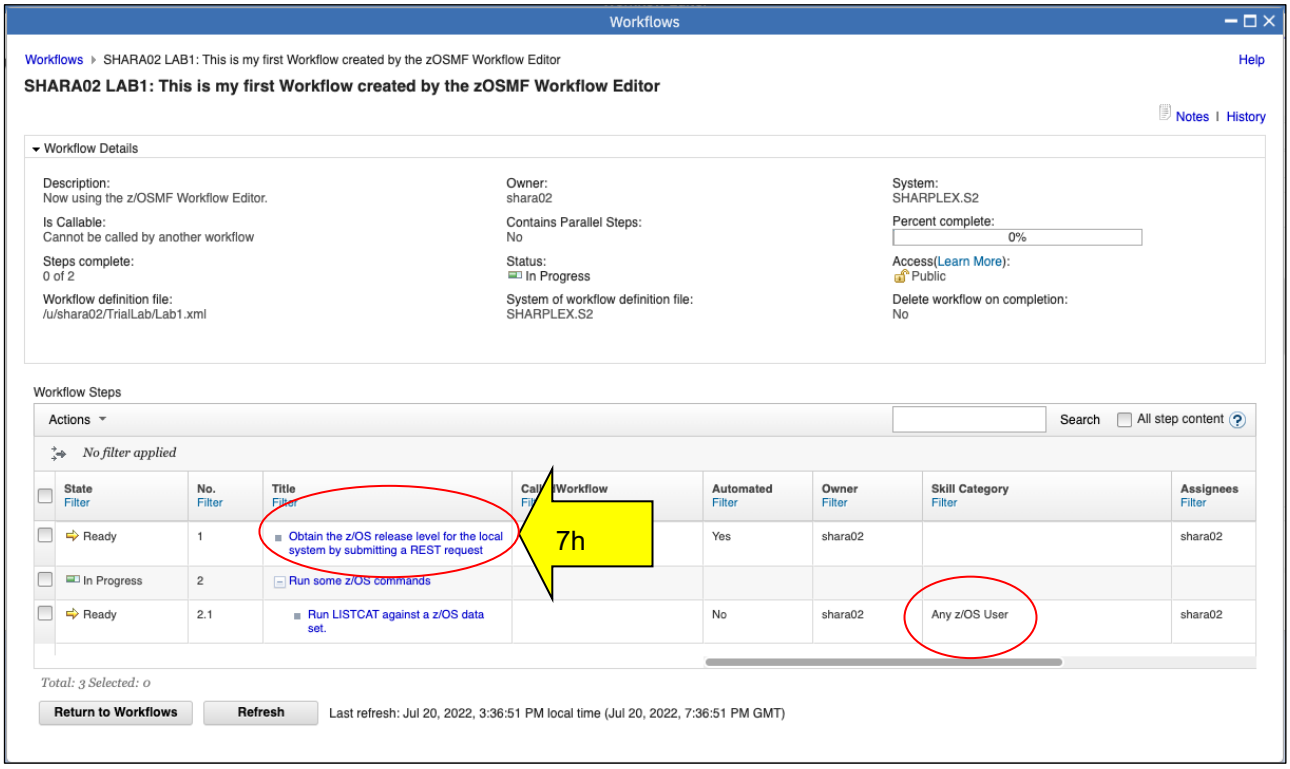
f. Click on **Assign all steps to owner user ID,** so that you can move the verification of this Workflow quickly. This is a shortcut we'll use to rapidly move into testing our Workflow.

g. Then click on **Finish.**



Now, this should look familiar. You can see a stand-alone leaf step called “Obtain the z/OS release level for the local system by submitting a REST request” and a Parent step call “Run some z/OS commands” with 1 leaf step called “Run LISTCAT against a z/OS data set.” (The Workflow Editor **Step Name** isn’t shown in the Workflow.) The Skill Category is what you put before (such as “Any z/OS user”, as shown below). These are the values that we used in the Editor called **Steps->Title** and **Steps->Skills.**

h. Click on the blue **Obtain the z/OS release level for the local system by submitting a REST request”,** so we can try out one of leaf steps we created.

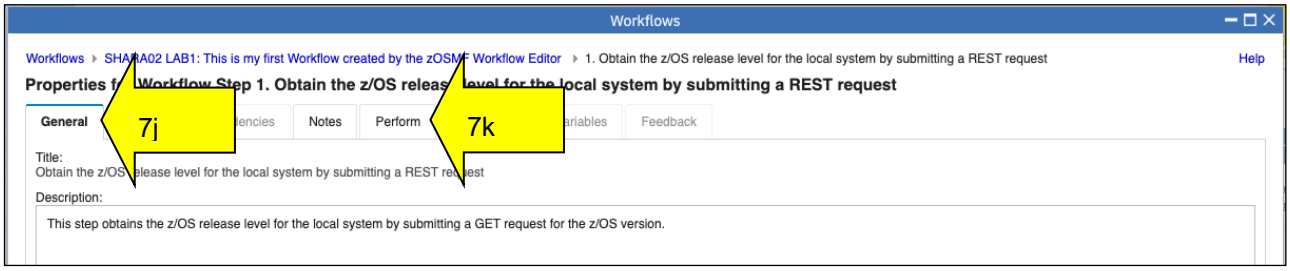


Notice that at any time you can refer back to your Workflow definition, by just clicking on the z/OSMF window for **Workflow Editor**, and then return to your Workflow by clicking back to the **Workflows** window. This is a handy way to compare what you put in the Editor, and what it looks like in a real Workflow for a user.

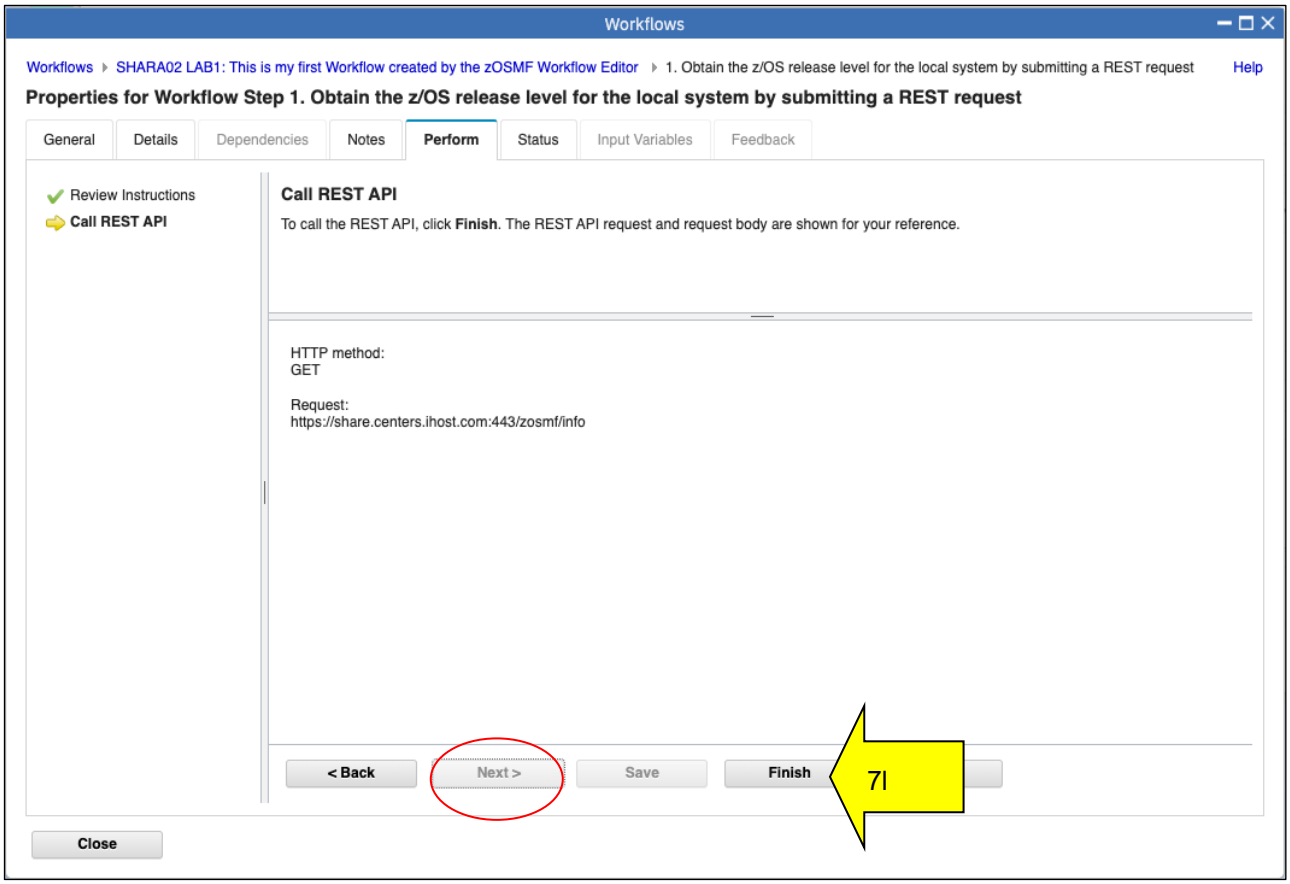
From the Workflow application, you can see the step information you provided from the Editor here.

j. Look at the information on the **General** tab (Editor input was: **Steps -> Step Details -> Step Overview -> Description)**

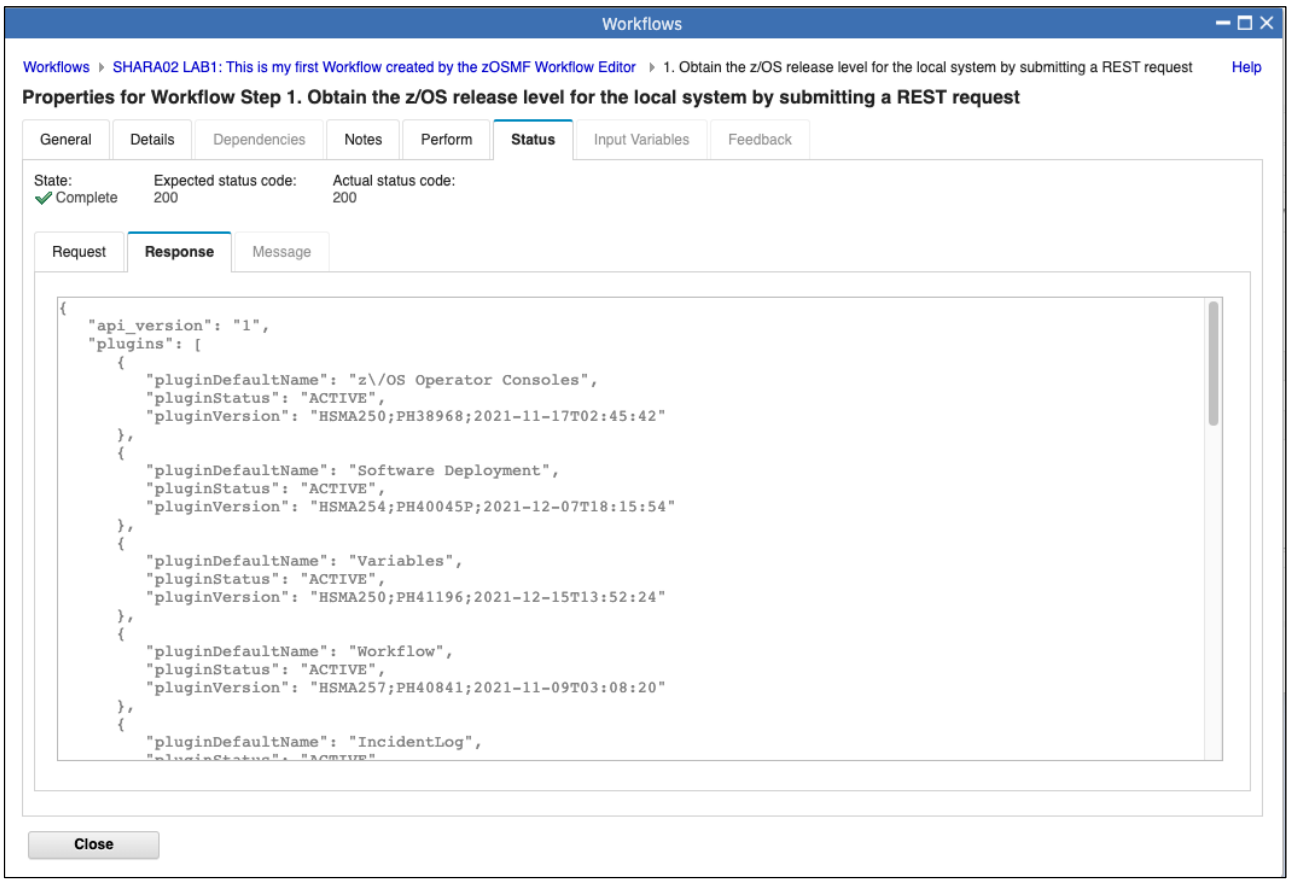
k. Go to the **Perform** tab.



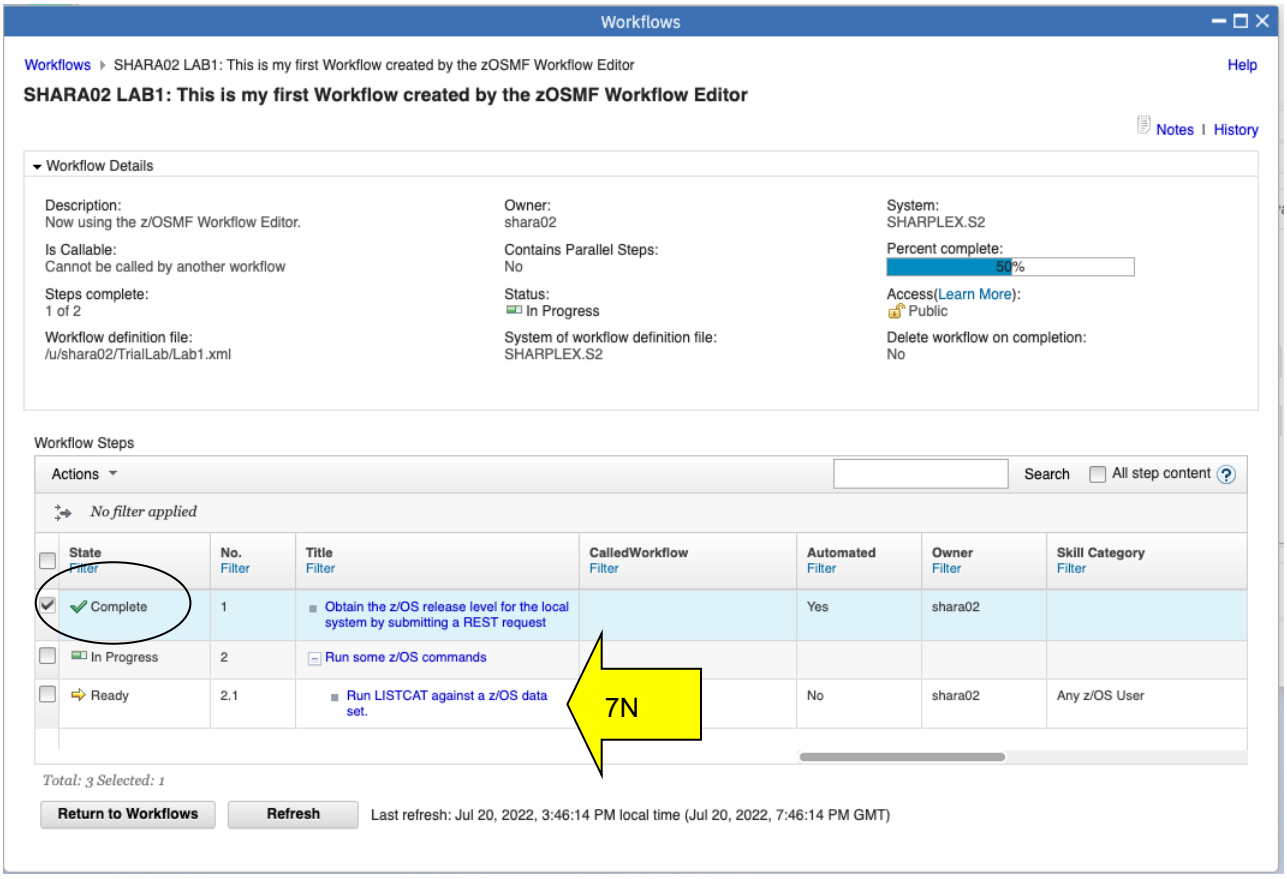
l. To Perform the step, since no variables are required you can keep hitting the "**Next**" until it is grayed out. Click "**Finish**" to execute the command.



m. Once the job finishes, you can look at the status to ensure that it executed fine. Click **Close**.



n. You should see that the step is Complete. Go ahead and try running the leaf step “Run LISTCAT against a z/OS data set.” . When prompted for a Data set to run LISTCAT against. You can use the data set **<YOUR ID>.ZFS.USER**. Ex: SHARA02.ZFS.USER



This concludes the most basic part of the lab. You have learned how to:

1. Create a brand-new Workflow
2. Import a step from the Step Library
3. Add variables to the Workflow, so that the user can provide unique information.
4. Add a parent step and a leaf step.
5. Associate the variables with the Workflow steps.
6. Run JCL in batch from a Workflow. (And how to code that JCL for the Velocity engine variables)
7. Save your Workflow definition file.
8. Know how to associate the Workflow Editor fields with what the user sees in a Workflow.

If you have enough time and want to continue, we will now proceed with… putting steps into a library for future import to other Workflows (i.e. reusing steps).

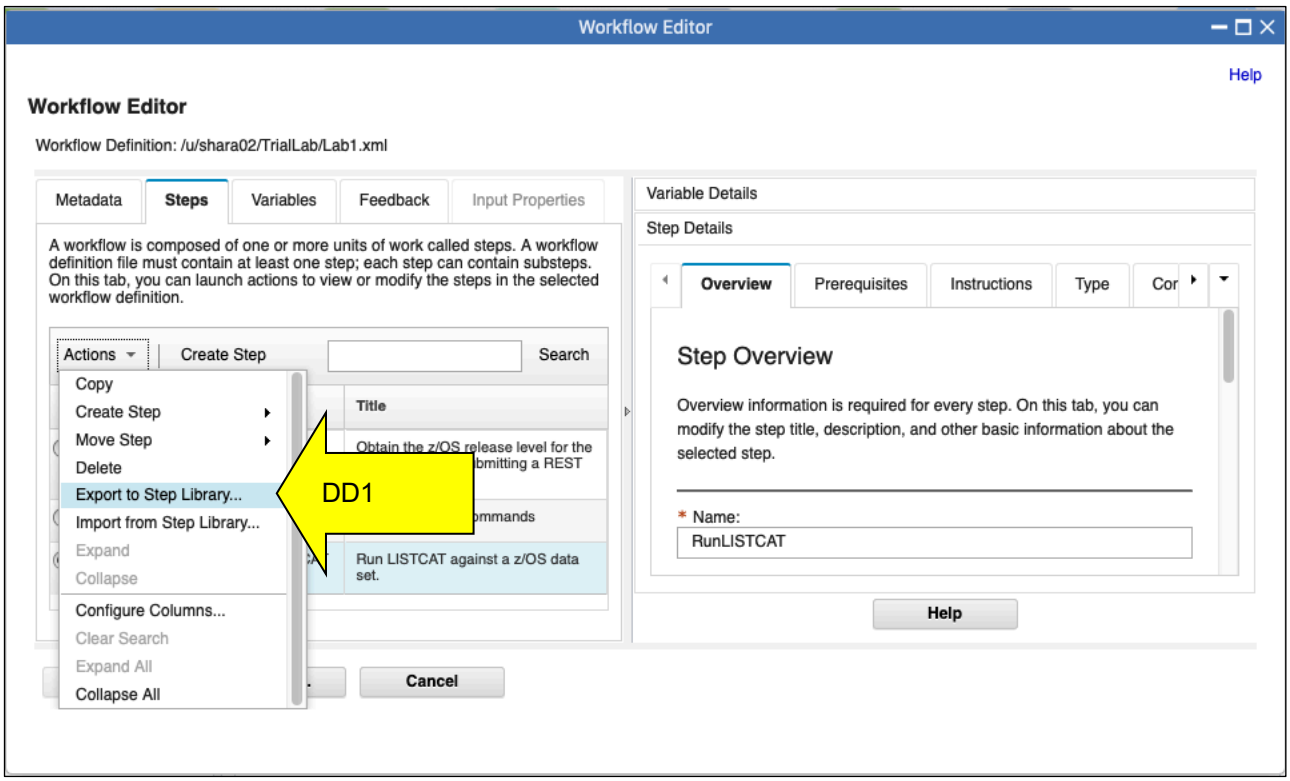
Deeper Dive

Putting steps into a library for future use in other Workflows

One of the nicest things available in the Workflow Editor, is that you can use the Editor to build a step, export it, and then import it into another Workflow you are building. This is a very nice way of reusing common steps that you need in several Workflows. We just saw how you could create a step that allocated a data set, which is a common activity. Let's now work on exporting that step, and then importing it into another Workflow.

DD1: Make sure you still are in the Editor and editing the Workflow definition file **/u/*YourUserid*/TrialLab/Lab1.xml.**

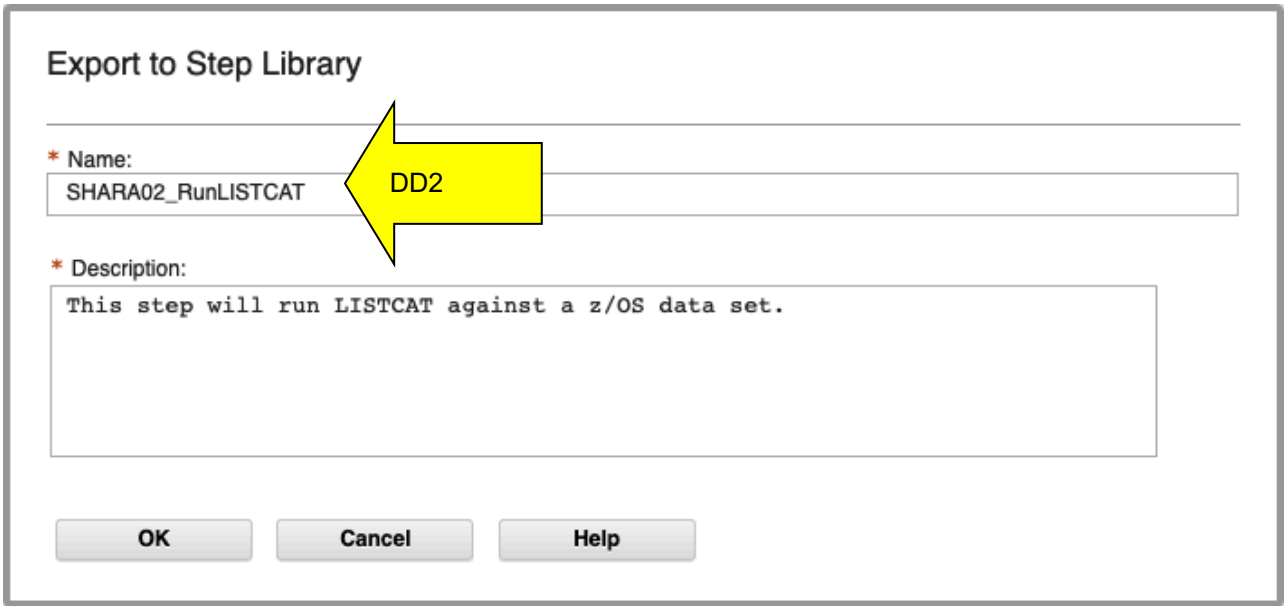
Click on **Steps**, and click on **step 2.1 (RunLISTCAT**) so we can export that step into our Step Library for other Workflows. Then click on **Actions** pulldown, and then select **Export to Step Library…**



DD2: You can now name the step in the library so others can find it. Because there are multiple people doing this lab, name the step something unique so you can find the specific one you export, and, so you don't have two steps in the same Workflow with the same name for this lab. Use something like ***YourUserid*_RunLISTCAT.** As we've seen before, the name cannot contain blanks.

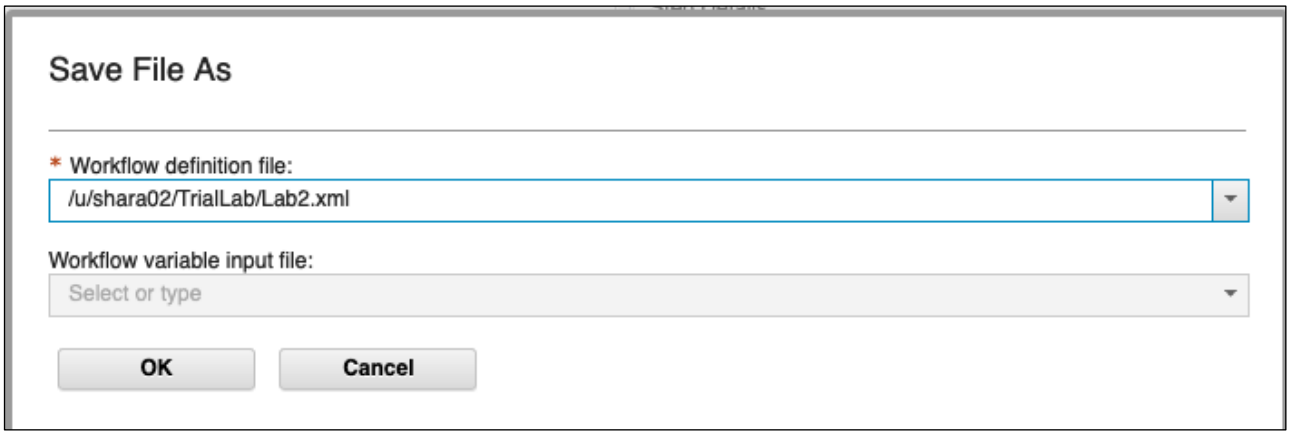
Also, type in a **Description** that you think will be helpful to other Workflow designers about what your exported step does.

Then click on **OK.**



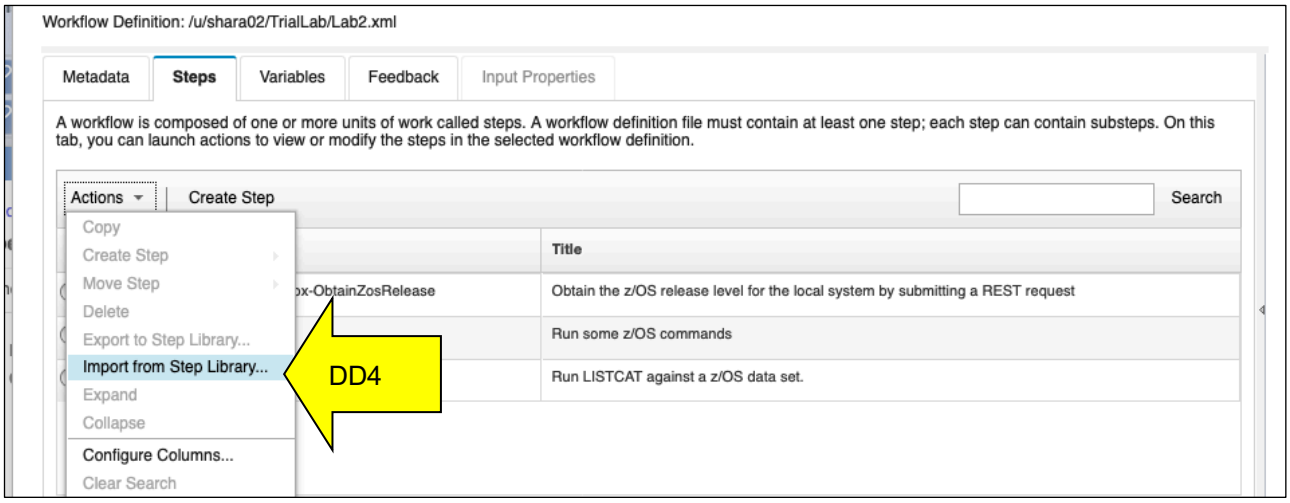
That was it! You have your step ready to share for others.

DD3: For the purposes of this lab, let's save our existing lab (Lab1.xml) into another Workflow definition file (called Lab2.xml) to perform our advance techniques on. Do a **Save As…** and type **/u/*YourUserID*/TrialLab/Lab2.xml**. Click on **OK.**



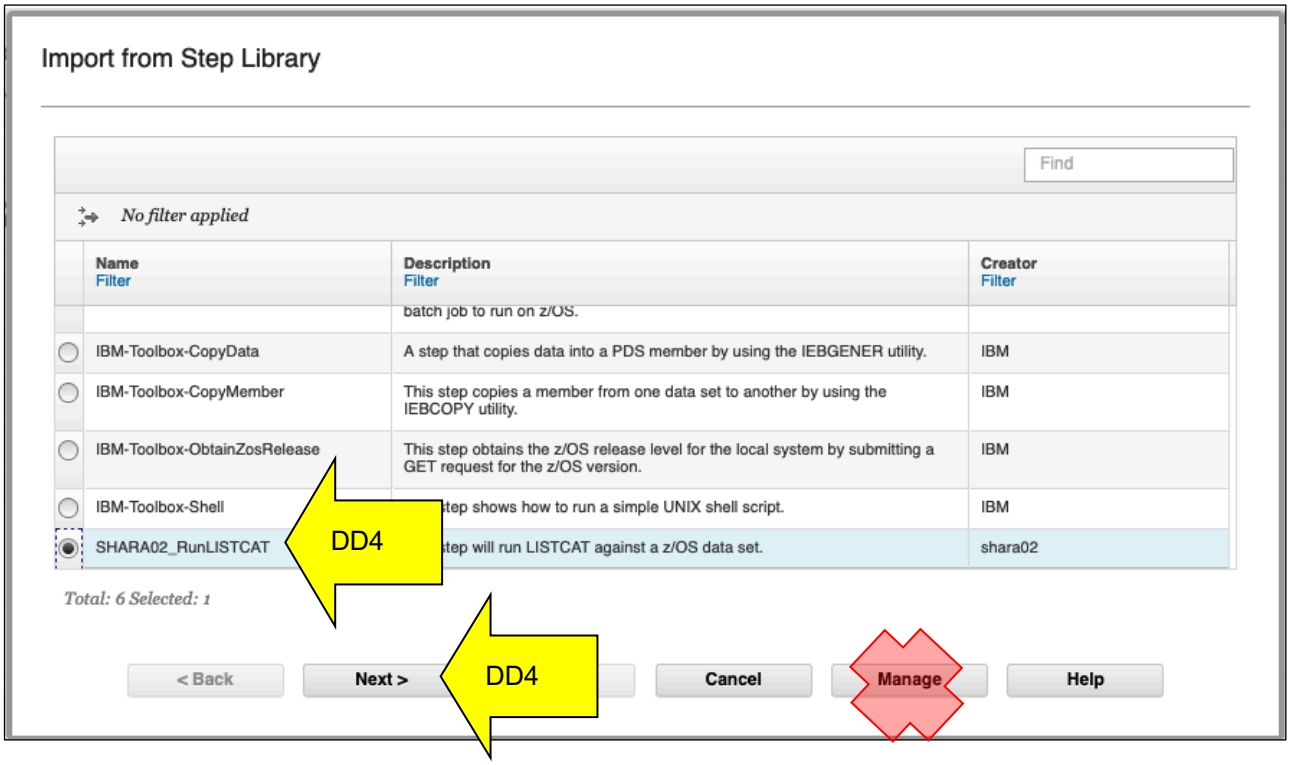
Verify that you saved into the right spot, so you are not losing or overlaying any of your work in Lab1.xml. Check the Fi**le Path:** name at the top to make sure you are *now* in **Lab2.xml.**

DD4: Under the Steps tab, click on Actions, Import from Step Library. (This should be familiar to you now)

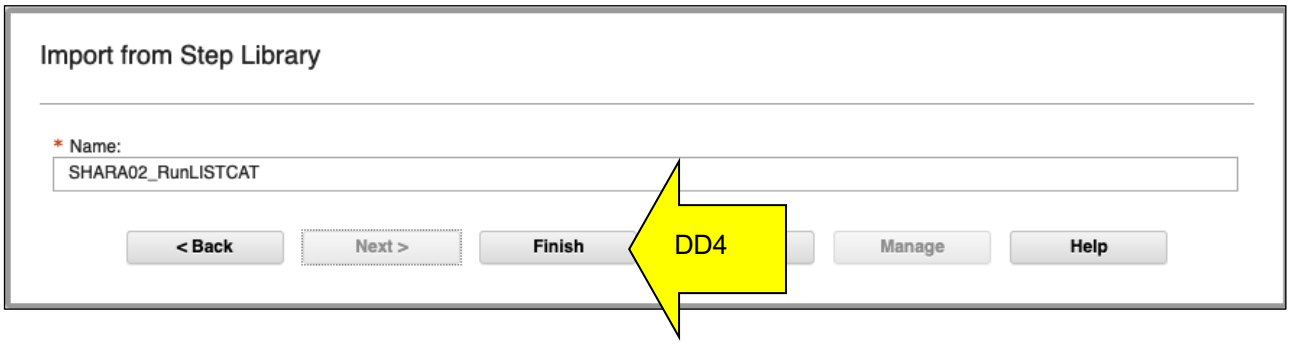


You can see the steps that are in the Step Library. There might be more than what is shown below, but you should notice your step that you exported above. Click on your exported step name (***YourUserID-_RunLISTCAT***), and then **Next.** If you don't see **Next,** you might have to resize the window.

As an aside point: if you wanted to delete a step from the Step Library, you *would* click on **Manage**, and then under Actions select **Delete** to delete your own steps.

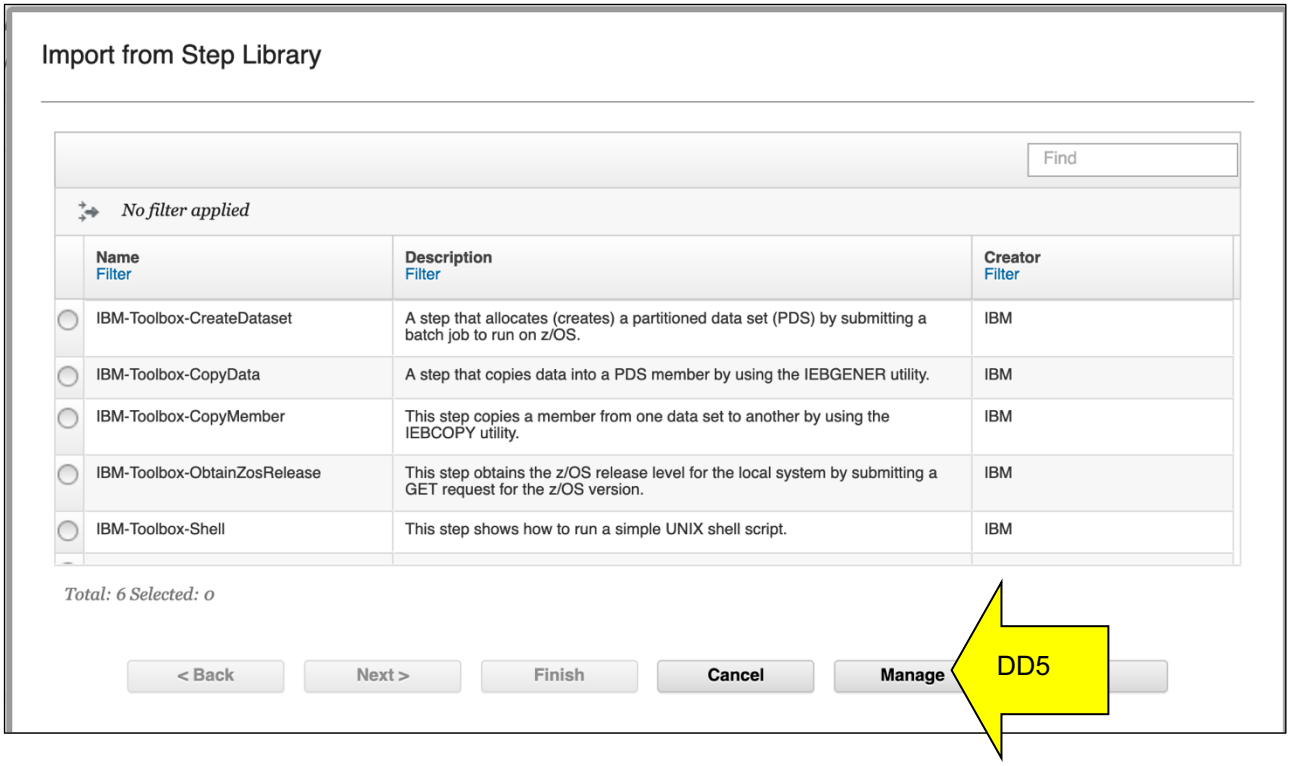


Then click on **Next>.** Which brings you to the follow. Click on **Finish.**

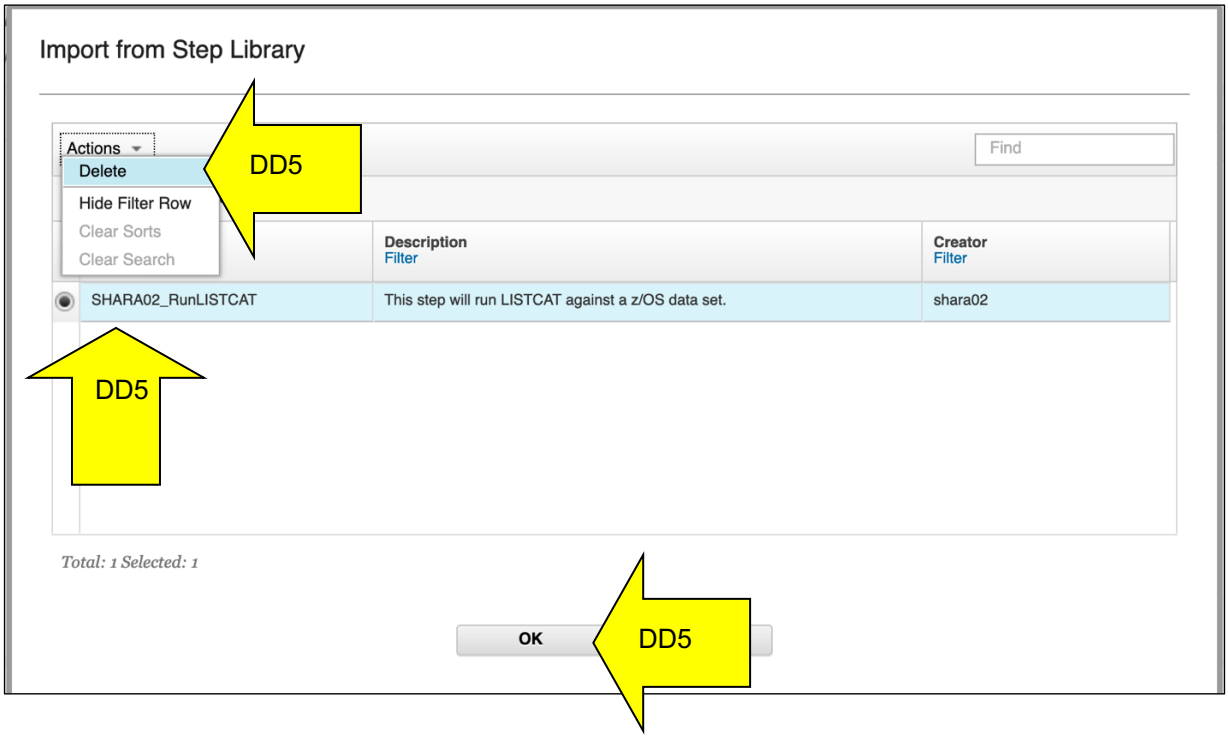


Success! Click on **Close.**

DD5: Now that we have added the step into our workflow, lets clean up the step we created from the Step Library. Go back into the Step Library (Actions -> Import from Step Library) and click **Manage**.



On the next window you should see the step you created. Go ahead and select that step, then click on **Actions** -> **Delete**. You will see the step removed.. Go ahead and click **OK**, and **Cancel** in the Step Library Window.



Take a step back...what happens when you first open a workflow definition?

Overview of Marshalling and Unmarshalling of Workflow files: When you open a workflow definition file for editing, the Workflow Editor locates the file and reads it into storage. The Workflow Editor also identifies any associated files that are referenced in the workflow definition and reads them into storage.

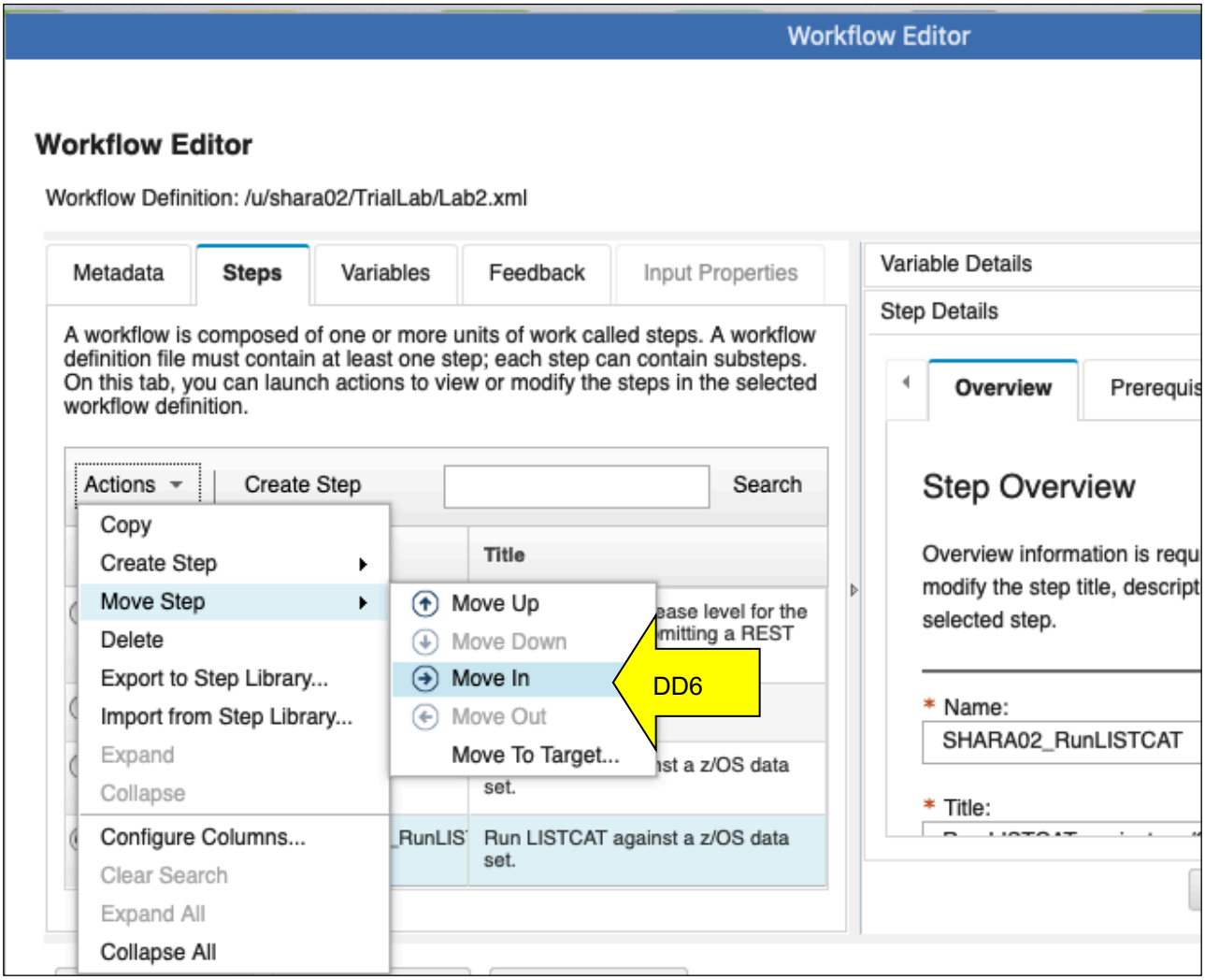
The Workflow Editor uses a process called *unmarshalling* to extract the contents of these files into its cache. Subsequently, when you save the edited file, the Workflow Editor uses a process called *marshalling* to create a single, consolidated workflow definition file that represents all of the requisite files. The resulting object is functionally equivalent to the original workflow structure, and the references to external files **are removed**.

During unmarshalling, there is a logical "break" in connection between the workflow definition and the external referenced XML files. The unmarshalling process creates a single workflow definition file as output.

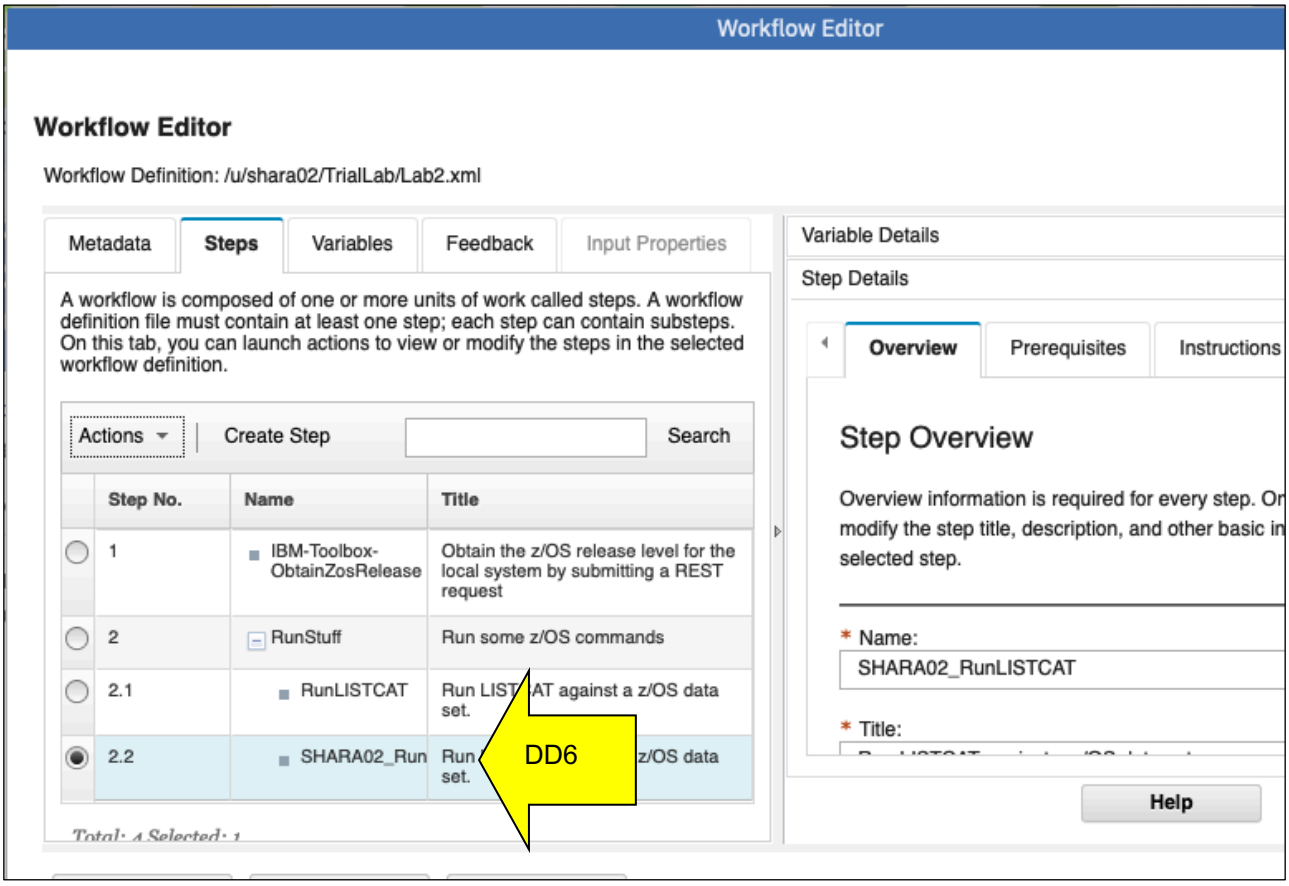
When a save is done, there is an update to the cached version of the workflow definition file, and also a write out of the XML to the file that was opened initially. Variables or step XML files that were referenced from the opened definition are not written out. This is where you should be aware of a “break”, as the changes are not written to a file template file.

Moving steps around

DD6: At this point, you have the step added after the parent step you selected. For this deep dive, let's pretend that we wanted this imported step to be prior parent step in our Workflow. It's easy to do, just click on the new step, **Actions,** then **Move Step >,** then **Move In.**

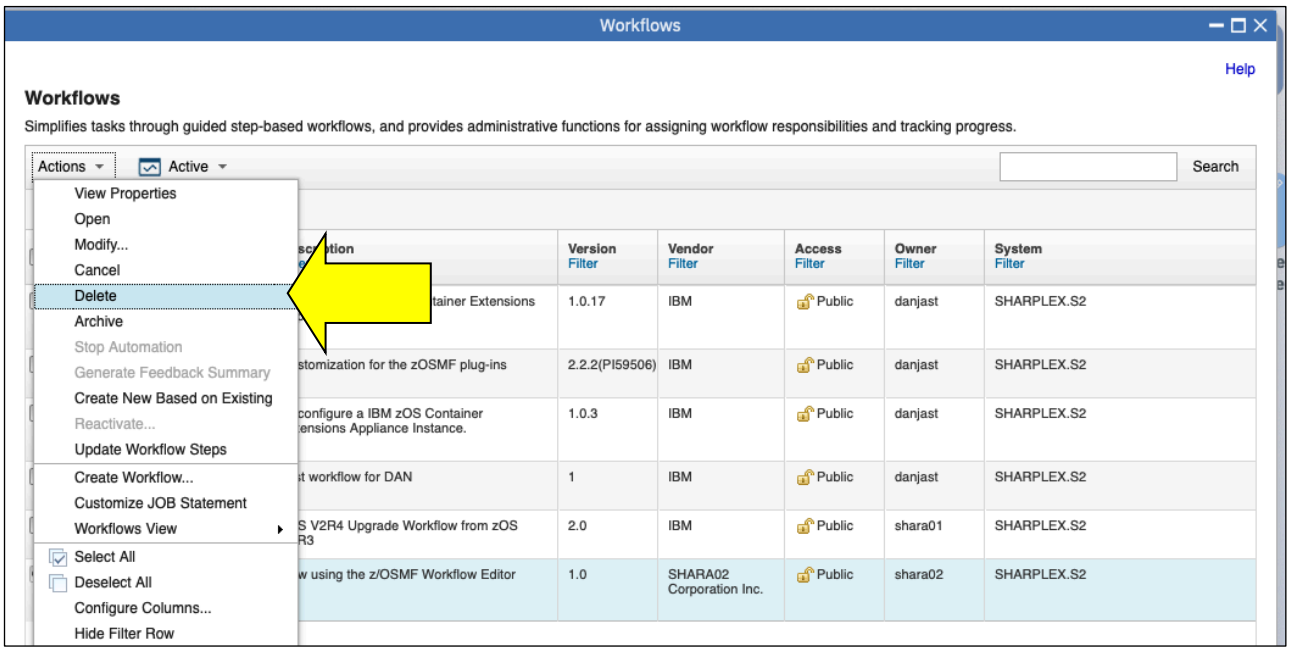


You can now see it's moved up into the parent step and is now Step 2.2 in the Workflow.



At the end of this exercise go ahead and delete the workflow you created.

Select the workflow you create. Then under **Actions** select **Delete**



Appendix of sample provided for this lab

Here is the sample of the **listcat** JCL. Notice all the **$instance-*variable*** locations for the Velocity engine.

```
VIEW       /u/mwalle/zosmf/labs/listcat                    Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 //LISTCAT EXEC PGM=IDCAMS
000002 //SYSPRINT DD SYSOUT=*
000003 //SYSIN    DD *
000004   LISTCAT ALL ENT($instance-DSN)
****** **************************** Bottom of Data ****************************
```