Hands-on Lab 2.2:
Orchestrating the Remaining Stages

Table of Contents

Objectives

In this HOL, you learn how to customize the Lab Management default template in order to orchestrate the acceptance test stage, the release stage, and the user acceptance test (UAT) stage of the release pipeline. The acceptance test stage is automatically triggered, while the other two stages are manually triggered. Orchestration for these stages includes:

- Naming each stage instance.
- Setting the location of the binaries repository.
- Ensuring that none of the stages build the binaries (builds only occur in the commit stage of the pipeline).
- Ensuring that the next stage retrieves the binaries if the current stage is successful.
- Stopping the pipeline if a stage fails.

This HOL is part two of the four-part Orchestration HOL. Together, the four parts demonstrate how to use Microsoft Team Foundation Server (TFS) and Lab Management to orchestrate the stages of a release pipeline that will support continuous delivery. The subject of orchestration is covered in Chapter 3 of Building a Release Pipeline with Team Foundation Server 2012.

The example application and services that are used in some exercises in this lab are in the subfolders of the **Lab02-Orchestration\Start-Lab** folder. Visual Studio solutions that are the result of completing all of the tasks in an exercise are in the **Lab02-Orchestration\Completed-Lab** folder. You run the examples for this lab on your local computer.

Prerequisites

Before you begin this lab you must first complete Lab2.1 – Orchestrating the Commit Stage.

Time

You should be able to complete this lab is about 40 minutes.

Exercise 1: Naming the Stage Instance and Setting the Binaries Repository Location

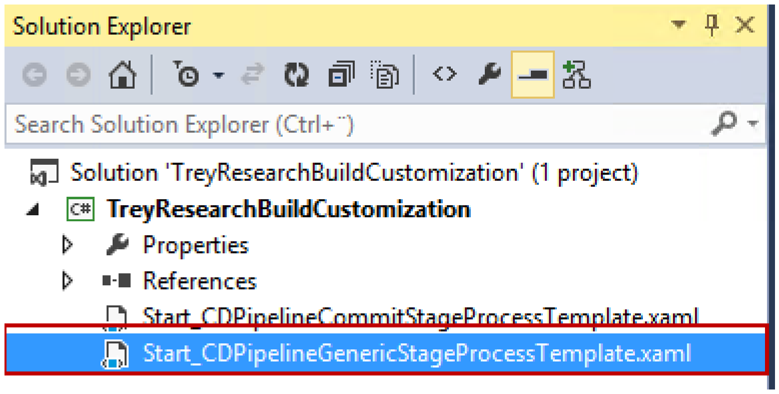
In this exercise, you modify a copy of the Lab Management template Start_CDPipelineGenericStageProcessTemplate.xaml, so that it can support orchestration. You customize the template to give the stage instance the same name as the pipeline instance. This name is a parameter. It is passed by the preceding stage of the pipeline if the current stage is automatically triggered. The parameter is provided by a user if the current stage is manually triggered.

You also customize the template to define the location of the binaries repository. The commit stage builds the binaries and stores them. The other stages need to know where to find those binaries. The repository location is also a parameter that is provided either by the preceding stage or by a user, depending on whether the current stage is triggered automatically or manually.

Task 1: Open the Solution in the Start-Lab Folder.

In this task you open the workflow file named Start_CDPipelineGenericStageProcessTemplate.xaml. This file is based on the LabDefaultTemplate.11.1.xaml file.

1. Navigate to **HOL\Lab02\Start-Lab**.
2. Open the **TreyResearchBuildCustomization.sln** file. You will see the following solution layout.

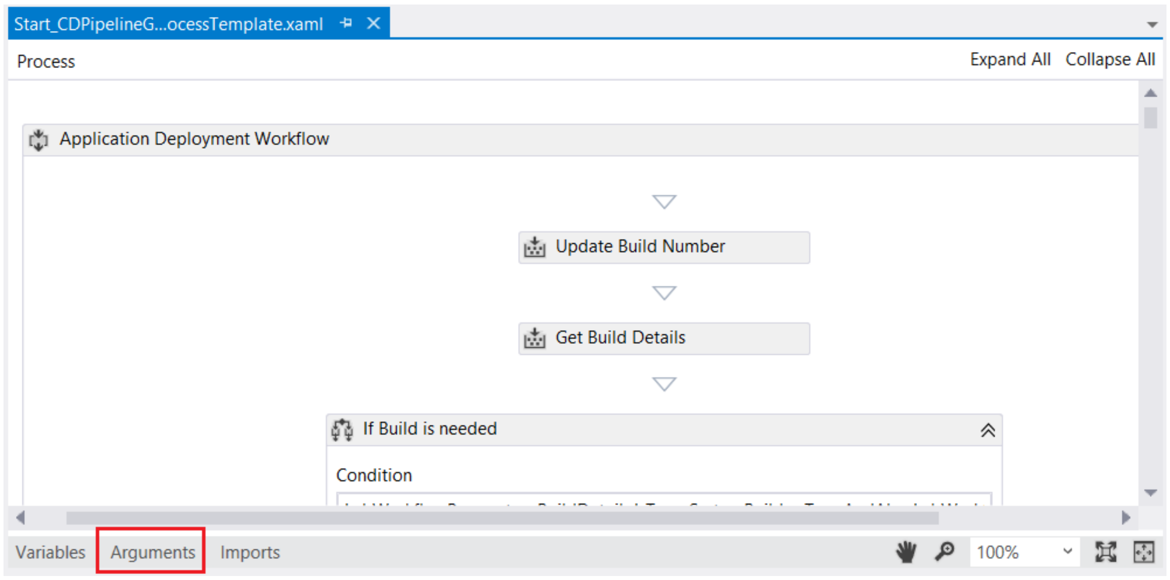

Solution Explorer
Search Solution Explorer (Ctrl+¨)
Solution 'TreyResearchBuildCustomization' (1 project)
TreyResearchBuildCustomization
Properties
References
Start_CDPipelineCommitStageProcessTemplate.xaml
Start_CDPipelineGenericStageProcessTemplate.xaml

3. In Visual Studio, open the **Start_CDPipelineGenericStageProcessTemplate.xaml** file. It opens in the workflow editor.

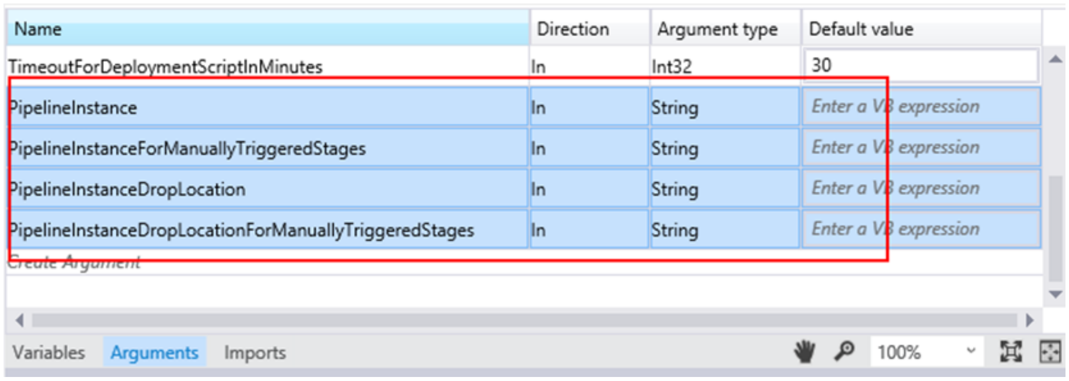
Task 2: Add Arguments to the Workflow

In this task you add arguments to the workflow that define the input parameters and the location of the binaries repository. You also modify the existing **BuildNumberFormat** argument.

1. Click the **Arguments** tab in the lower-left corner of the workflow editor. The **Arguments** pane opens.

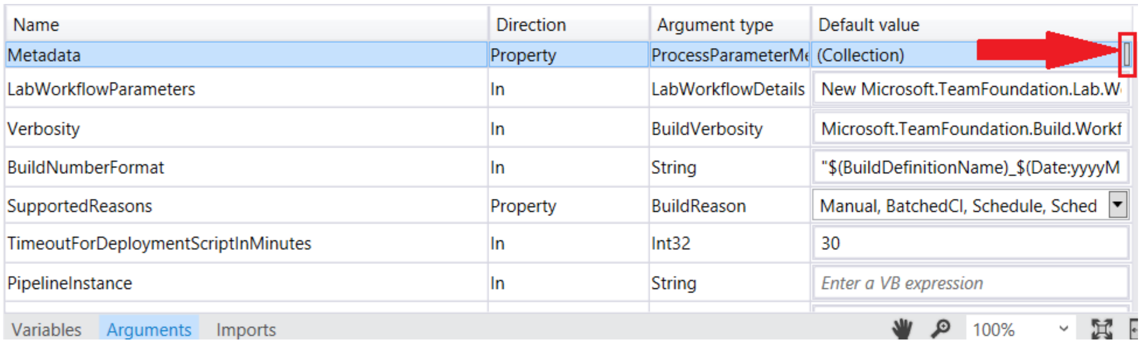


2. In the **Arguments** pane, click ***Create Argument***. In the **Name** column enter **PipelineInstance**.
3. Click ***Create Argument***. In the **Name** column enter **PipelineInstanceForManuallyTriggeredStages**.
4. Click ***Create Argument***. In the **Name** column enter **PipelineInstanceDropLocation**.
5. Click ***Create Argument***. In the **Name** column enter **PipelineInstanceDropLocationForManuallyTriggeredStages**.



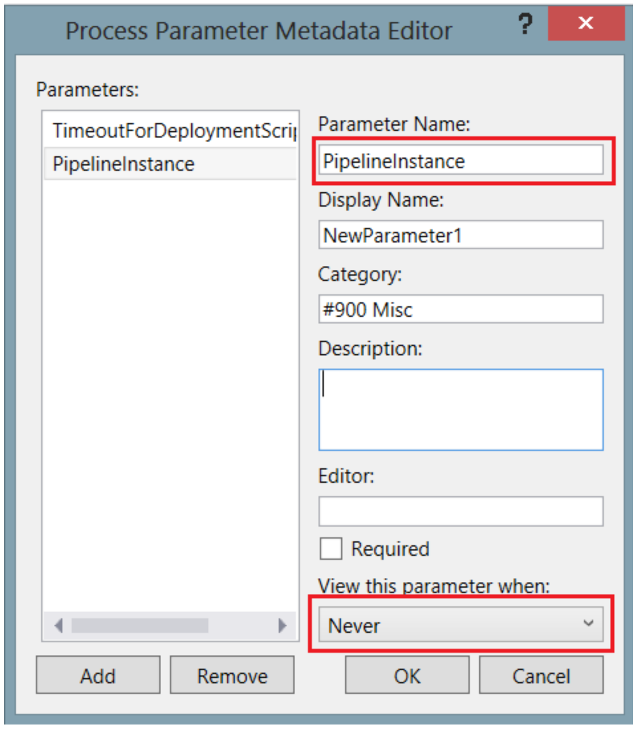
Name	Direction	Argument type	Default value
TimeoutForDeploymentScriptInMinutes	In	Int32	30
PipelineInstance	In	String	*Enter a VB expression*
PipelineInstanceForManuallyTriggeredStages	In	String	*Enter a VB expression*
PipelineInstanceDropLocation	In	String	*Enter a VB expression*
PipelineInstanceDropLocationForManuallyTriggeredStages	In	String	*Enter a VB expression*

6. Click the **Edit** button in the **Default value** column of the **Metadata** row.



Name	Direction	Argument type	Default value
Metadata	Property	ProcessParameterMe	(Collection)
LabWorkflowParameters	In	LabWorkflowDetails	New Microsoft.TeamFoundation.Lab.W
Verbosity	In	BuildVerbosity	Microsoft.TeamFoundation.Build.Workf
BuildNumberFormat	In	String	"$(BuildDefinitionName)_$(Date:yyyyM
SupportedReasons	Property	BuildReason	Manual, BatchedCI, Schedule, Sched
TimeoutForDeploymentScriptInMinutes	In	Int32	30
PipelineInstance	In	String	Enter a VB expression

7. The **Process Parameter Metadata Editor** opens. Click **Add**. Enter the **PipelineInstance** metadata that is highlighted in red in the following screenshot.

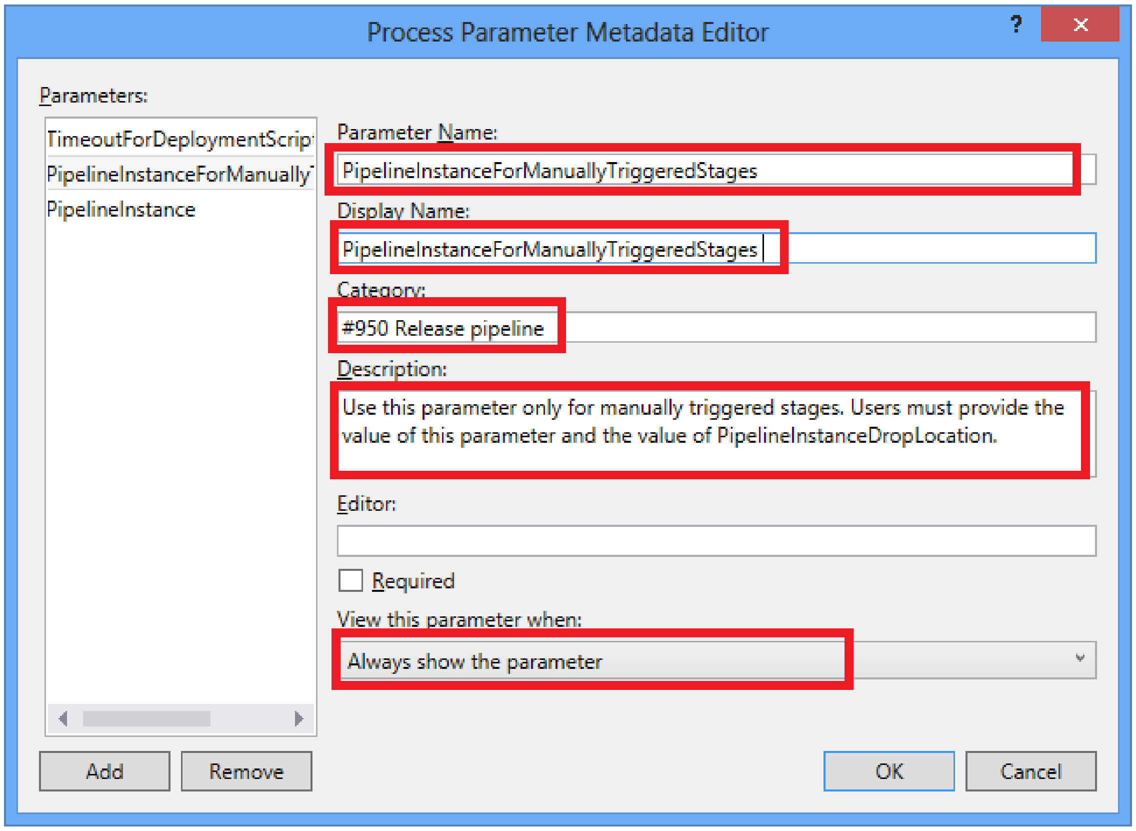


Process Parameter Metadata Editor

Parameters:
TimeoutForDeploymentScri
PipelineInstance

Parameter Name: PipelineInstance
Display Name: NewParameter1
Category: #900 Misc
Description:
Editor:
Required
View this parameter when: Never

8. Click **Add**. Enter the **PipelineInstanceForManuallyTriggeredStages** metadata that is highlighted in red in the following screenshot.


Process Parameter Metadata Editor
Parameters:
TimeoutForDeploymentScrip
PipelineInstanceForManually
PipelineInstance
Add
Remove
OK
Cancel

Parameter Name: PipelineInstanceForManuallyTriggeredStages

Display Name: PipelineInstanceForManuallyTriggeredStages

Category: #950 Release pipeline

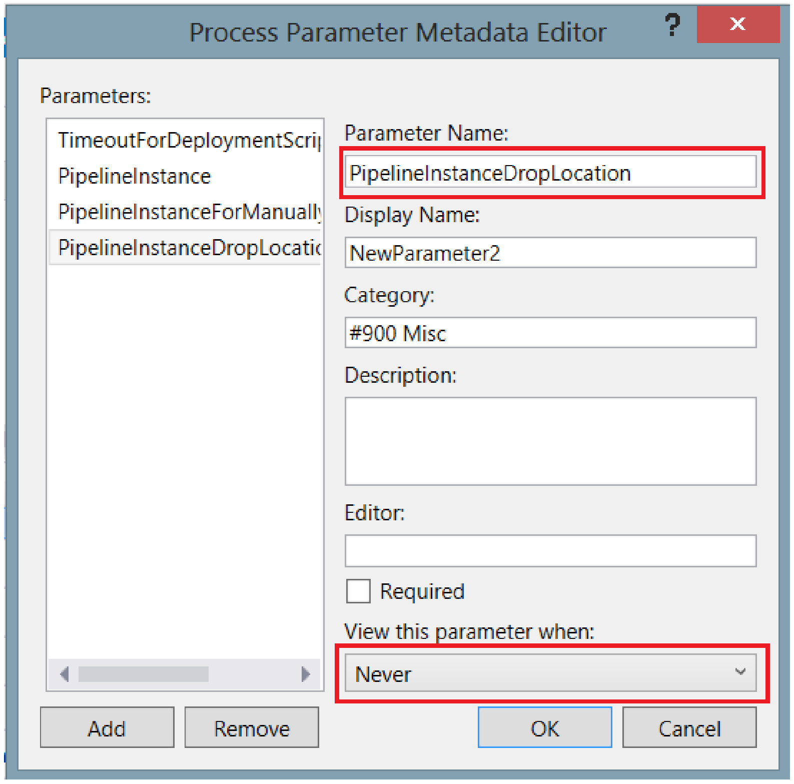
Description: Use this parameter only for manually triggered stages. Users must provide the value of this parameter and the value of PipelineInstanceDropLocation.

Editor:

☐ Required

View this parameter when: Always show the parameter

9. Click **Add**. Enter the **PipelineInstanceDropLocation** metadata that is highlighted in red in the following screenshot.


Process Parameter Metadata Editor

Parameters:

- TimeoutForDeploymentScri
- PipelineInstance
- PipelineInstanceForManuall
- PipelineInstanceDropLocatio

Parameter Name: PipelineInstanceDropLocation

Display Name: NewParameter2

Category: #900 Misc

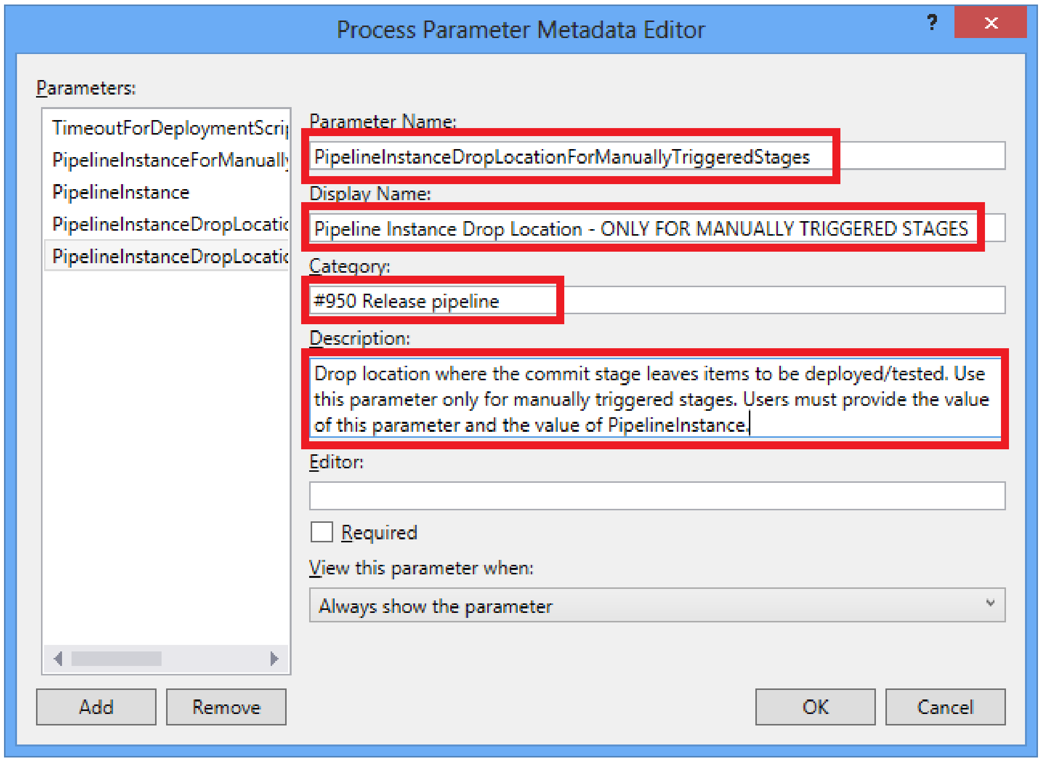
Description:

Editor:

Required: ☐

View this parameter when: Never

10. Click **Add**. Enter the **PipelineInstanceDropLocationForManuallyTriggeredStages** metadata that is highlighted in red in the following screenshot.


Process Parameter Metadata Editor

Parameters:

- TimeoutForDeploymentScri
- PipelineInstanceForManuall
- PipelineInstance
- PipelineInstanceDropLocatio
- PipelineInstanceDropLocatio

Parameter Name: PipelineInstanceDropLocationForManuallyTriggeredStages

Display Name: Pipeline Instance Drop Location - ONLY FOR MANUALLY TRIGGERED STAGES

Category: #950 Release pipeline

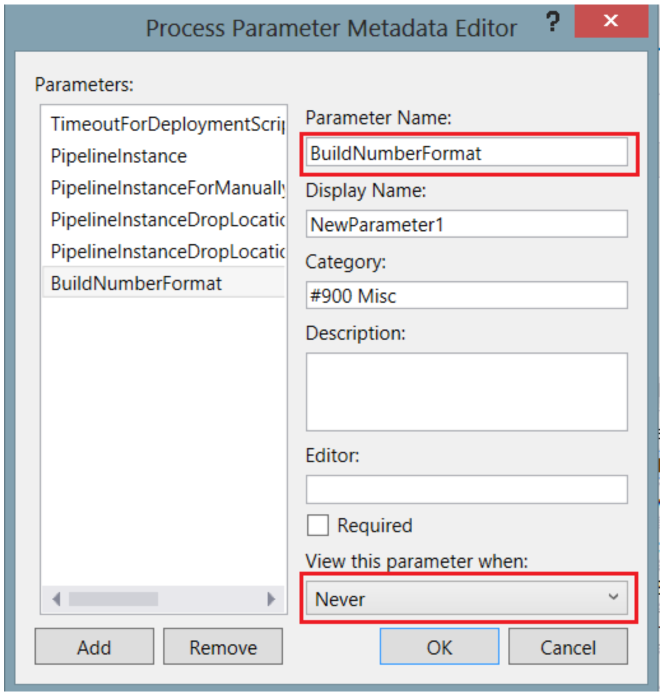
Description: Drop location where the commit stage leaves items to be deployed/tested. Use this parameter only for manually triggered stages. Users must provide the value of this parameter and the value of PipelineInstance.

Editor:

Required: ☐

View this parameter when: Always show the parameter

11. Click **Add**. Enter the **BuildNumberFormat** metadata that is highlighted in red in the following screenshot.


Process Parameter Metadata Editor

Parameters:

TimeoutForDeploymentScrip
PipelineInstance
PipelineInstanceForManually
PipelineInstanceDropLocatic
PipelineInstanceDropLocatic
BuildNumberFormat

Parameter Name: BuildNumberFormat

Display Name: NewParameter1

Category: #900 Misc

Description:

Editor:

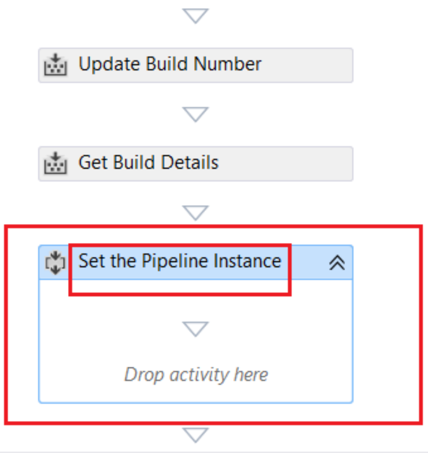
Required

View this parameter when: Never

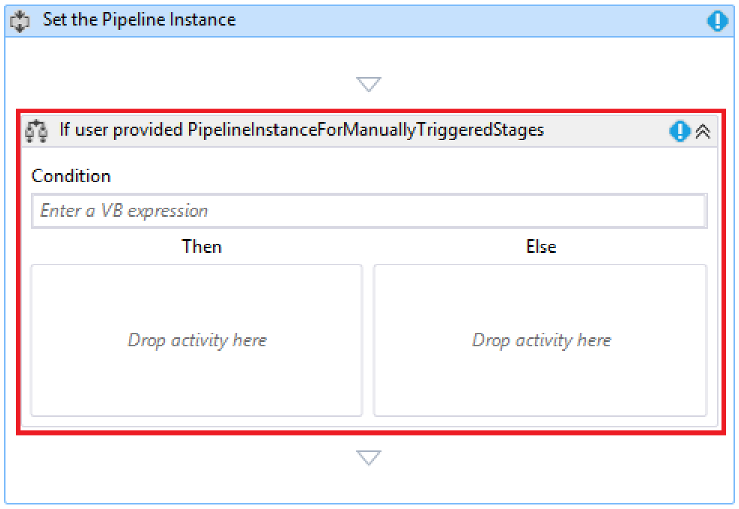
Task 3: Add the Set the Pipeline Instance Sequence

In this task you add a new sequence that sets the stage name. How the name is set is determined by whether the stage is triggered automatically or manually.

1. Go to the beginning of the workflow and locate the **Get Build Details** activity.
2. Add a **Sequence** after the **Get Build Details** activity and name it **Set the Pipeline Instance**.



3. Add an **If** activity to the sequence and name it **If user provided PipelineInstanceForManuallyTriggeredStages**. (The "!" symbol occurs in the following screenshot because the properties aren't filled in yet.)

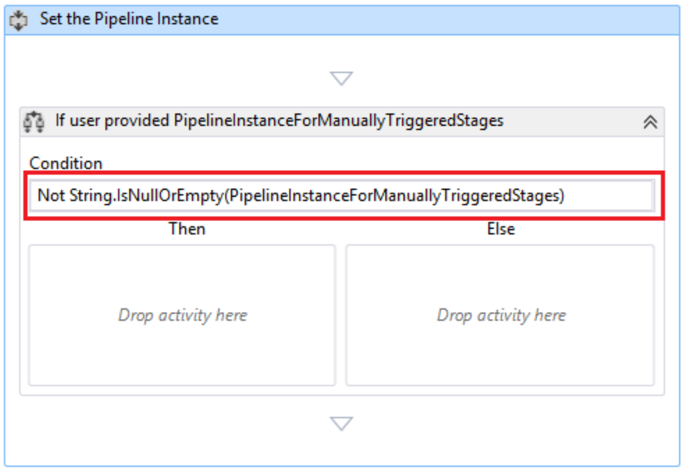

Set the Pipeline Instance
If user provided PipelineInstanceForManuallyTriggeredStages
Condition
Enter a VB expression
Then
Else
Drop activity here
Drop activity here

4. Set the **Condition** to check if the stage is manually triggered by adding the following code.

Visual Basic

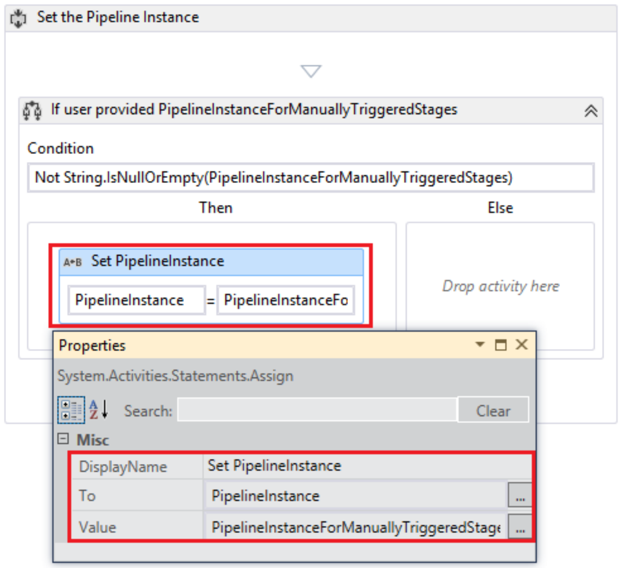
```
Not String.IsNullOrEmpty(PipelineInstanceForManuallyTriggeredStages)
```

Here is a screenshot of the **If** activity with the **Condition** filled in.


Set the Pipeline Instance
If user provided PipelineInstanceForManuallyTriggeredStages
Condition
Then
Else
Drop activity here
Drop activity here

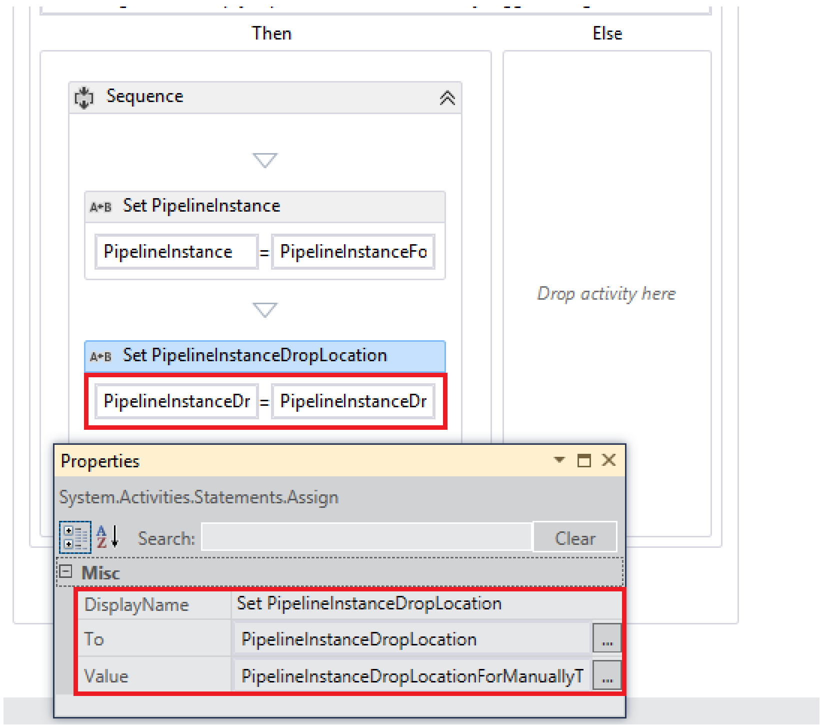
Condition: Not String.IsNullOrEmpty(PipelineInstanceForManuallyTriggeredStages)

5. Add an **Assign** activity to the **Then** box and name it **Set PipelineInstance**. For manually triggered stages, this activity replaces the **PipelineInstance** variable with the **PipelineInstanceForManuallyTriggeredStages** variable.


Set the Pipeline Instance
If user provided PipelineInstanceForManuallyTriggeredStages
Condition
Not String.IsNullOrEmpty(PipelineInstanceForManuallyTriggeredStages)
Then
Else
Set PipelineInstance
PipelineInstance = PipelineInstanceFo
Drop activity here
Properties
System.Activities.Statements.Assign
Search:
Clear
Misc

DisplayName	Set PipelineInstance
To	PipelineInstance
Value	PipelineInstanceForManuallyTriggeredStage

6. Add another **Assign** activity to the **Then** box and name it **Set PipelineInstanceDropLocation**. For manually triggered stages, this activity replaces the **PipelineInstanceDropLocation** variable with the **PipelineInstanceDropLocationForManuallyTriggeredStages** variable.

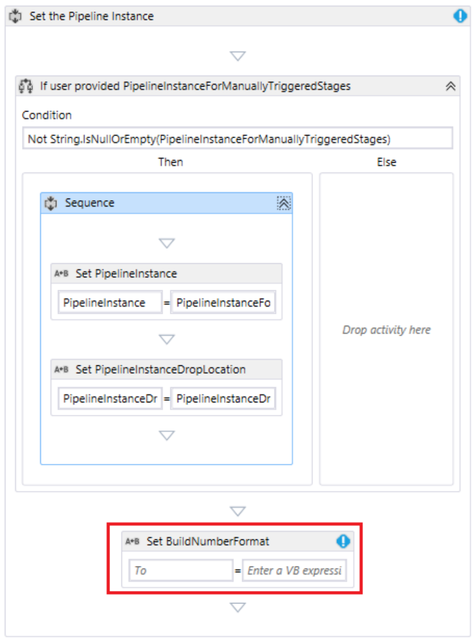

Then
Else
Sequence
Set PipelineInstance
PipelineInstance = PipelineInstanceFo
Set PipelineInstanceDropLocation
PipelineInstanceDr = PipelineInstanceDr
Drop activity here
Properties
System.Activities.Statements.Assign
Search:
Clear
Misc

DisplayName	Set PipelineInstanceDropLocation
To	PipelineInstanceDropLocation
Value	PipelineInstanceDropLocationForManuallyT

Task 4: Set the BuildNumberFormat

In this task you set the build number format.

1. Add an **Assign** activity after the **If** activity and name it **Set BuildNumberFormat**.


Set the Pipeline Instance

If user provided PipelineInstanceForManuallyTriggeredStages

Condition

Not String.IsNullOrEmpty(PipelineInstanceForManuallyTriggeredStages)

Then

Else

Sequence

Set PipelineInstance

PipelineInstance = PipelineInstanceFo

Set PipelineInstanceDropLocation

PipelineInstanceDr = PipelineInstanceDr

Drop activity here

Set BuildNumberFormat

To = Enter a VB expressi

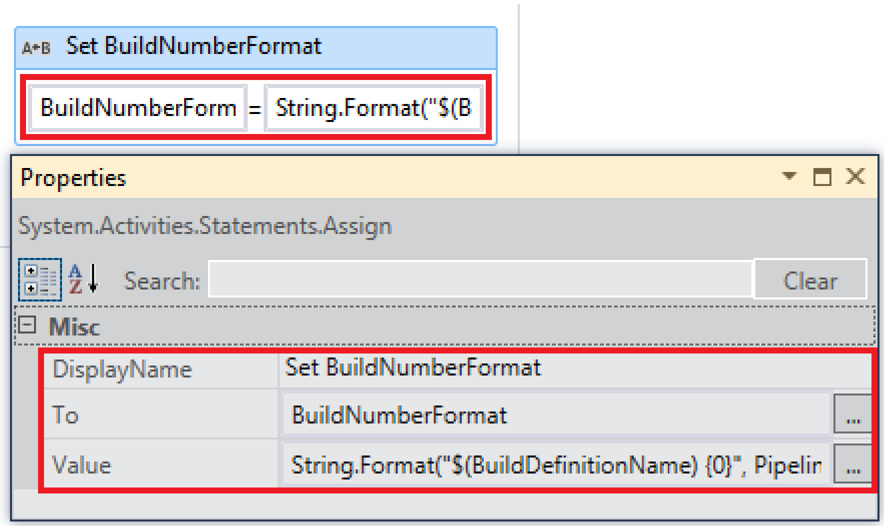
2. Set the **Set BuildNumberFormat** activity's **To** argument to **BuildNumberFormat**.

3. Set the **Set BuildNumberFormat** activity's **Value** argument to the following code.

Visual Basic

```
String.Format("$(BuildDefinitionName) {0}", PipelineInstance)
```

Here is a screenshot of the **Assign** activity properties with the values filled in.


Set BuildNumberFormat

BuildNumberForm = String.Format("$(B

Properties

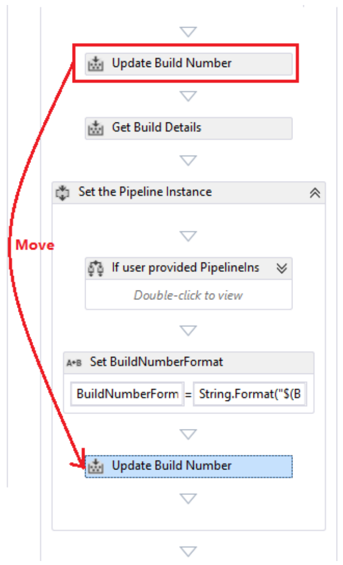
System.Activities.Statements.Assign

Search: Clear

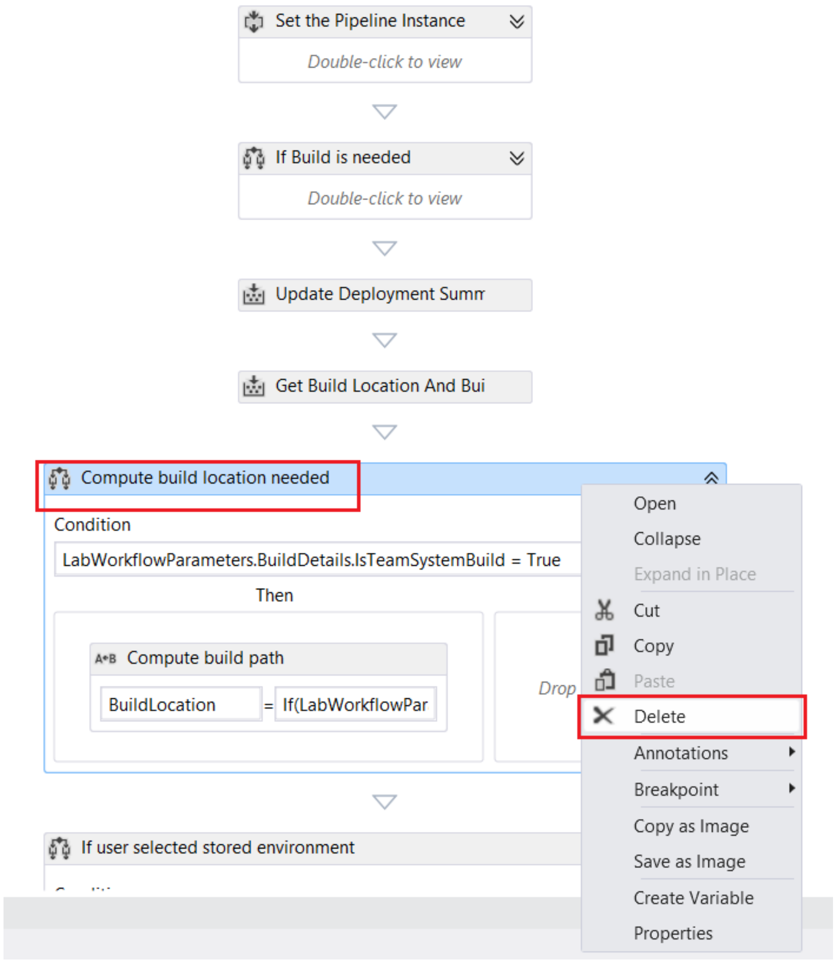
Misc

DisplayName	Set BuildNumberFormat
To	BuildNumberFormat
Value	String.Format("$(BuildDefinitionName) {0}", Pipelir

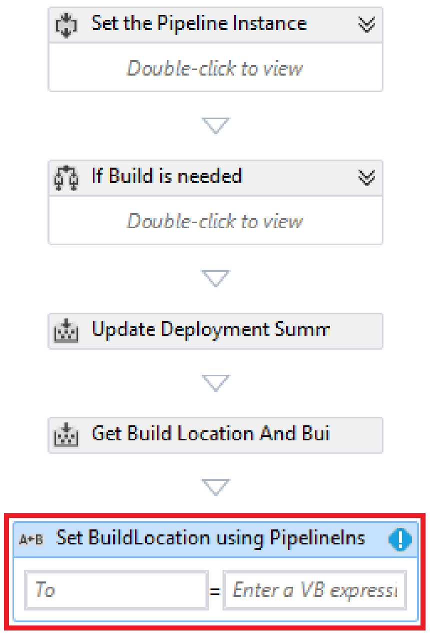
4. Move the **Update Build Number** activity that's located at the very top of the workflow to just below the **Set BuildNumberFormat** activity. You do this so that the build number is updated with the proper value and before any other activities start to run.



5. If you move three activities down the workflow from your current location, you will find an activity named **Compute build location needed**. Delete it because this activity already occurs earlier in the stage.



6. Replace it with an **Assign** activity and name it **Set BuildLocation using PipelineInstanceDropLocation**.


Set the Pipeline Instance

Double-click to view

If Build is needed

Double-click to view

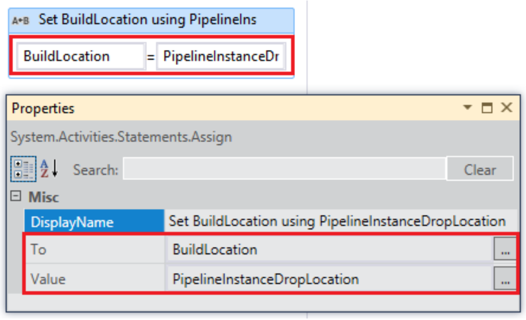
Update Deployment Summ

Get Build Location And Bui

Set BuildLocation using PipelineIns

To = Enter a VB expressi

7. Set the **Assign** activity's **To** argument to **BuildLocation**.
8. Set the **Value** argument for the **BuildLocation** parameter to **PipelineInstanceDropLocation**.


Set BuildLocation using PipelineIns

BuildLocation = PipelineInstanceDr

Properties

System.Activities.Statements.Assign

Search: Clear

Misc

DisplayName	Set BuildLocation using PipelineInstanceDropLocation
To	BuildLocation
Value	PipelineInstanceDropLocation

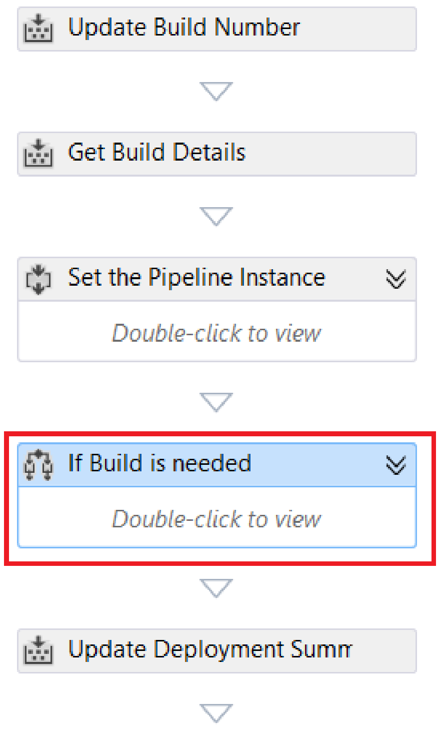
Exercise 2: Building Only Once

In this exercise you make certain that there are no redundant builds because a continuous delivery pipeline only builds the binaries once. To do this, you remove the build step from the workflow. Nothing is built in any stage other than the commit stage, even if a user requests it. Unlike the **BuildNumberFormat** parameter, the build step can't be hidden from users. Make sure they know that the build step is disabled.

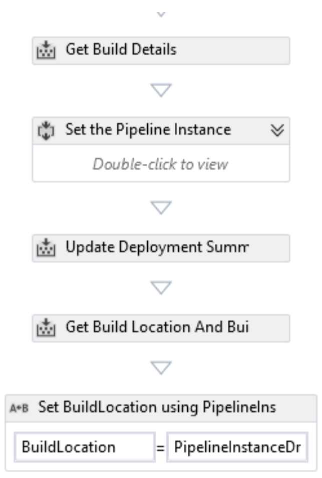
Task 1: Locate and Replace the Build Activity

In this task you locate and replace an activity named **If Build is needed**.

1. Locate the **If Build is needed** activity. It is below the **Set the Pipeline Instance** activity.



2. Right-click on the **If Build is needed** activity and delete it. The workflow should now have the **Set the Pipeline Instance** activity followed by the **Update Deployment Summary** activity.



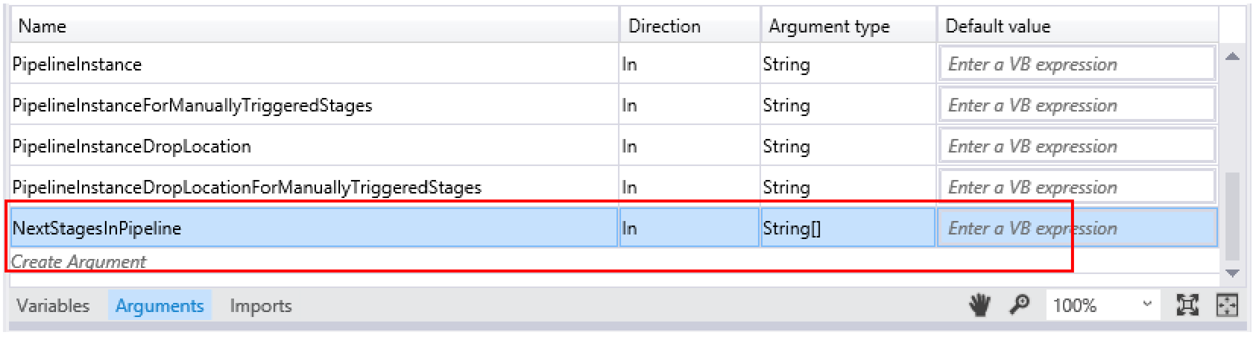
Exercise 3: Propagating Changes Automatically or Stopping the Pipeline

In this exercise you configure the workflow to either trigger the next stage if the current stage succeeds, or to stop the pipeline if the current stage fails. The exercise is similar to exercise 3, task 2 in Lab 2.1.

Task 1: Add the NextStagesInPipeline Argument

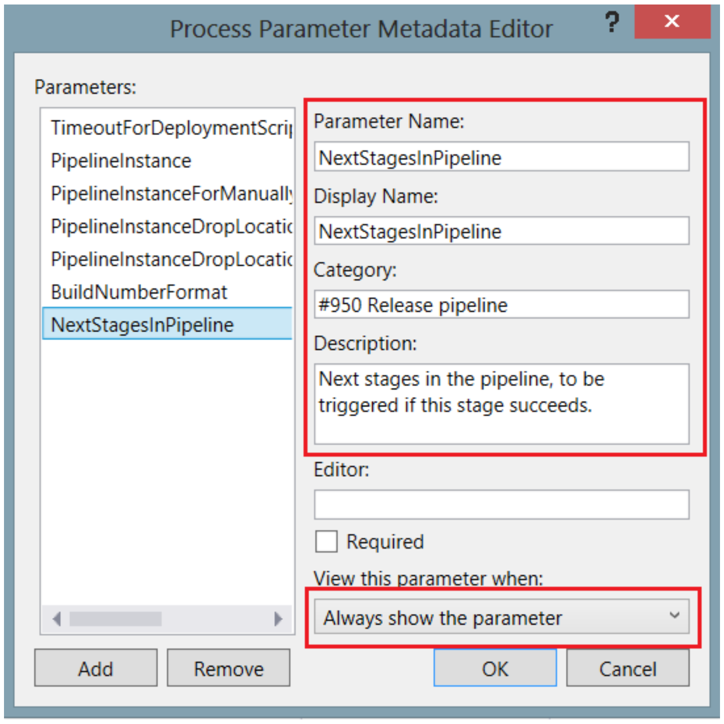
In this task you add a new argument and modify its parameters.

1. In the workflow editor, click **Arguments**. The **Arguments** pane appears. Click ***Create Argument***. Add a new argument named **NextStagesInPipeline**. Its argument type is **String[]**.



Name	Direction	Argument type	Default value
PipelineInstance	In	String	Enter a VB expression
PipelineInstanceForManuallyTriggeredStages	In	String	Enter a VB expression
PipelineInstanceDropLocation	In	String	Enter a VB expression
PipelineInstanceDropLocationForManuallyTriggeredStages	In	String	Enter a VB expression
NextStagesInPipeline	In	String[]	Enter a VB expression
Create Argument			

2. Open the **Process Parameter Metadata Editor**. Click **Add**. Enter the **NextStagesInPipeline** metadata that is highlighted in red in the following screenshot.



Process Parameter Metadata Editor

Parameters:

TimeoutForDeploymentScri
PipelineInstance
PipelineInstanceForManuall
PipelineInstanceDropLocatic
PipelineInstanceDropLocatic
BuildNumberFormat
NextStagesInPipeline

Parameter Name: NextStagesInPipeline

Display Name: NextStagesInPipeline

Category: #950 Release pipeline

Description: Next stages in the pipeline, to be triggered if this stage succeeds.

Editor:

Required

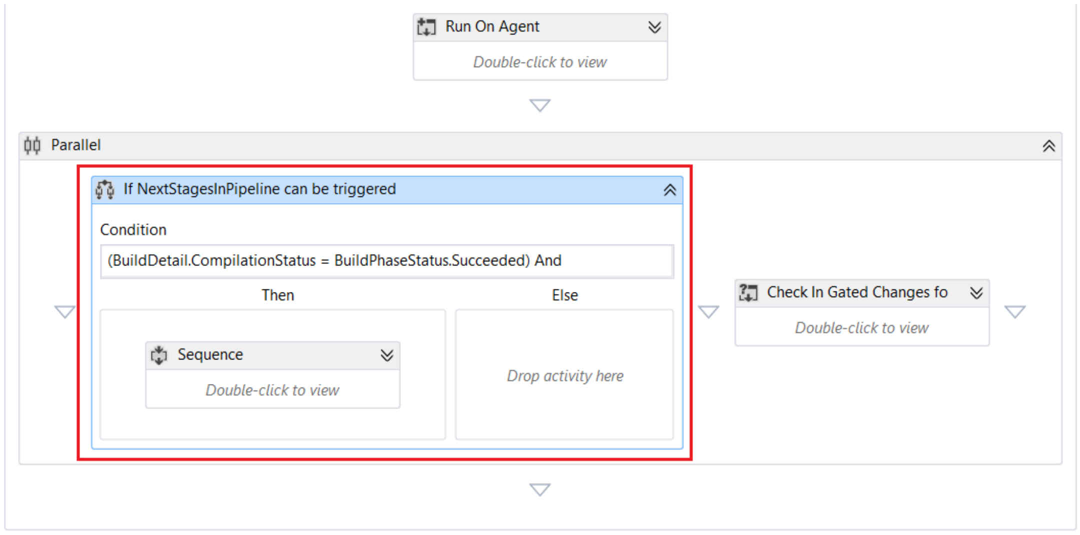
View this parameter when: Always show the parameter

Task 2: Add If NextStagesInPipeline can be triggered Activity

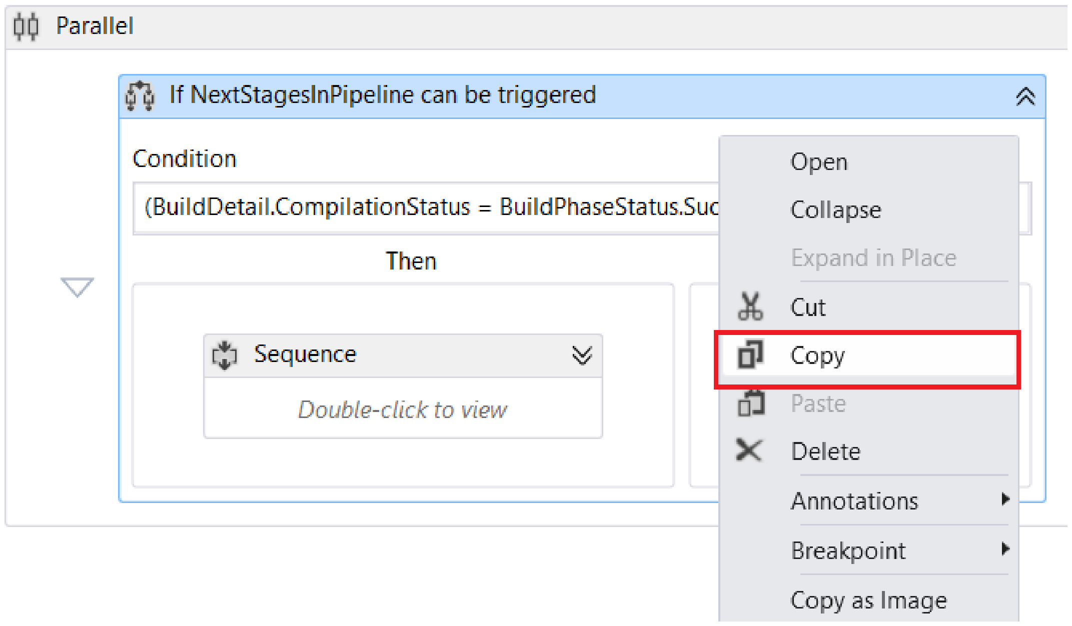
This task is a repeat of exercise 4 of Lab2.1. You will reuse the results of that exercise here.

1. Open the Start_CDPipelineCommitStageProcessTemplate.xaml file that you edited in Lab2.1.

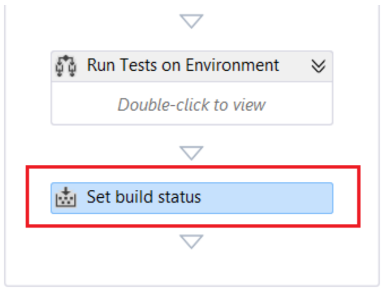
2. Scroll to end of the file and locate the **If NextStagesInPipeline can be triggered** activity.


Run On Agent
Double-click to view
Parallel
If NextStagesInPipeline can be triggered
Condition
(BuildDetail.CompilationStatus = BuildPhaseStatus.Succeeded) And
Then
Else
Sequence
Double-click to view
Drop activity here
Check In Gated Changes fo
Double-click to view

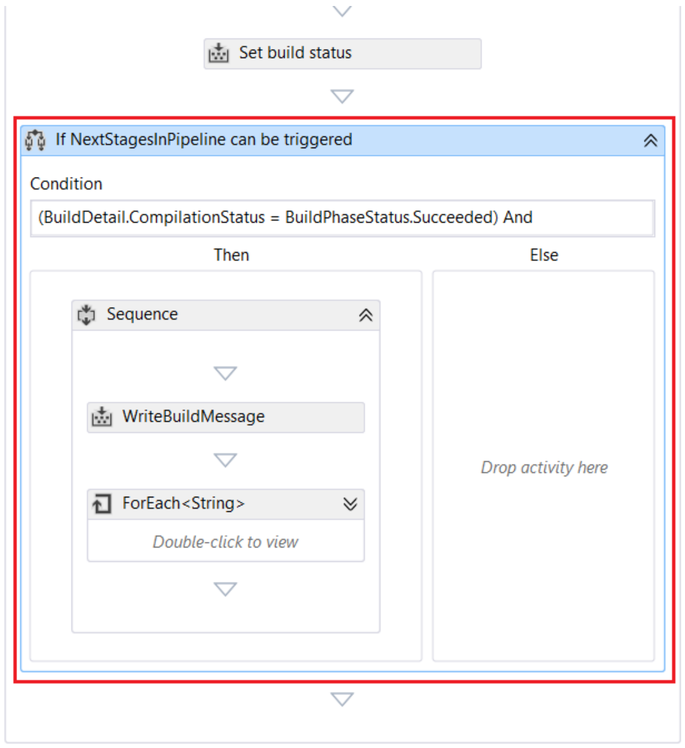
3. Right-click the **If NextStagesInPipeline can be triggered** activity and copy it.



4. Go back to the Start_CDPipelineGenericStageProcessTemplate.xaml file.
5. Scroll to the end of the file and locate the **Set build status** activity.



6. Place your cursor after the **Set build status** activity, right-click and paste the copied **If NextStagesInPipeline can be triggered** activity.



(BuildDetail.CompilationStatus = BuildPhaseStatus.Succeeded) And

Summary

In this HOL you customized the Lab Management default template so that you could orchestrate the remaining stages of the release pipeline. At this point, only the commit stage runs any steps that are not directly related to orchestration. For example, the commit stage performs continuous integration and code analysis. Steps in the other stages are currently placeholders.

Copyright