LAB MANUAL

Automation Boot Camp

N-able RMM

Version 2.11

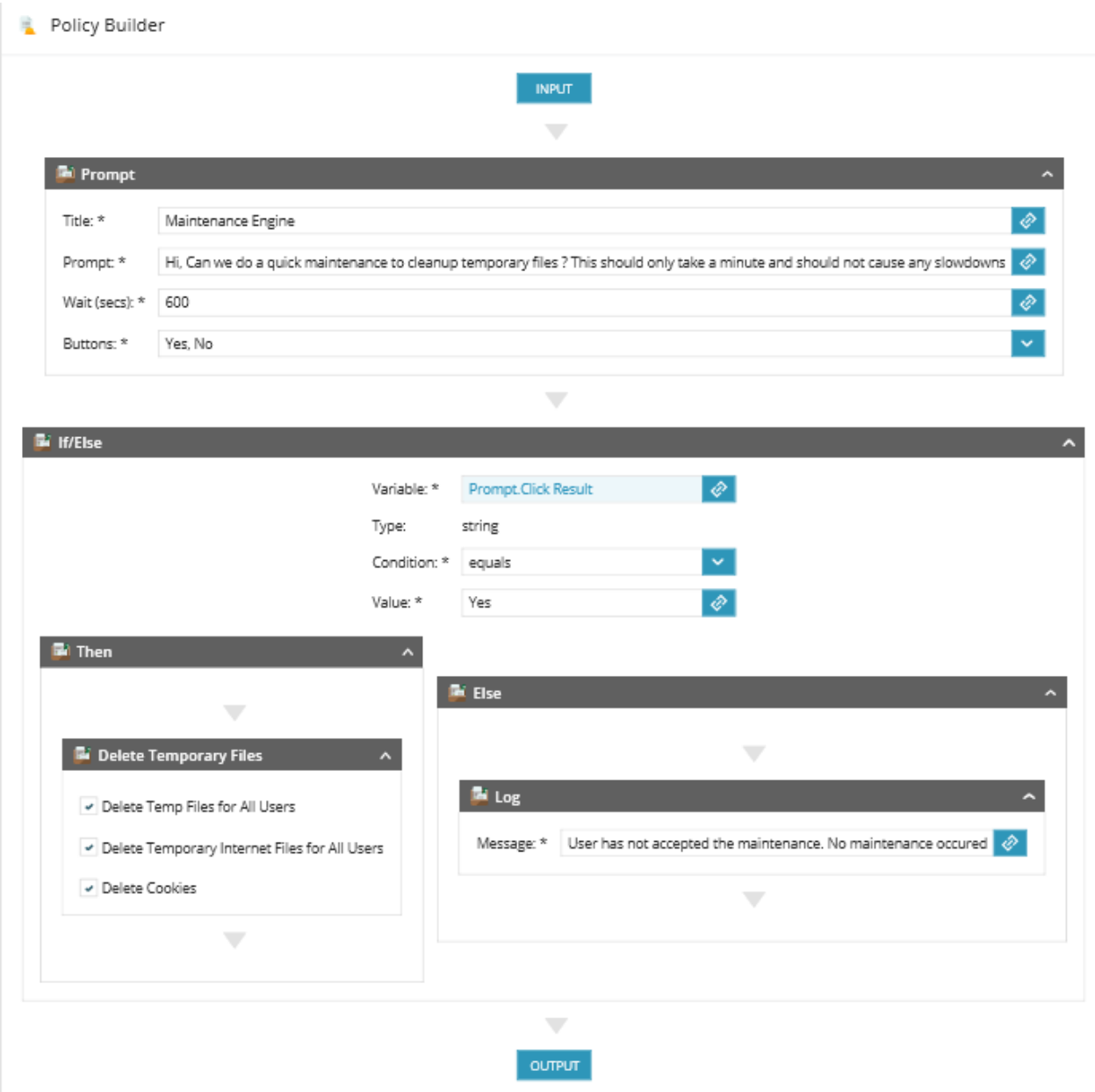
N-ABLE™

Last updated:

October 29, 2021

Table of Contents

Lab 1 – Self-healing maintenance



Objective

Create a basic disk clean-up policy to free space on a user's disk.

Estimated time

10 minutes

Required resources

Automation manager application

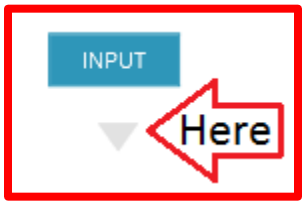
Self-guided summary (advanced version)

For this lab, you will create an automation policy called **LAB 01 - <*YOURNAME*>**.

The goal is to create a policy that prompts the user for permission, and based on their answer, either runs the **Disk Cleanup** policy or not.

Detailed steps

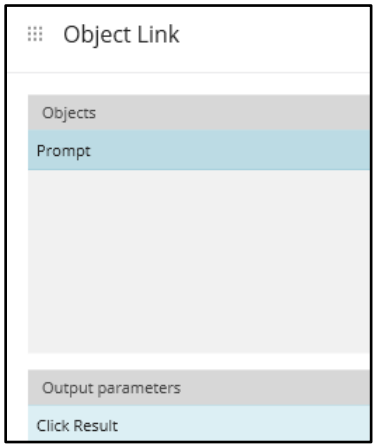
1. Create a new policy in automation manager
    a. Name it **LAB 01 - <*YOURNAME*>**, and describe it as "Lab 01 - Disk Cleanup".
2. Add the required objects:
    a. In the left menu, search for the **Prompt** object and drag it to the middle of the **Policy Builder**:



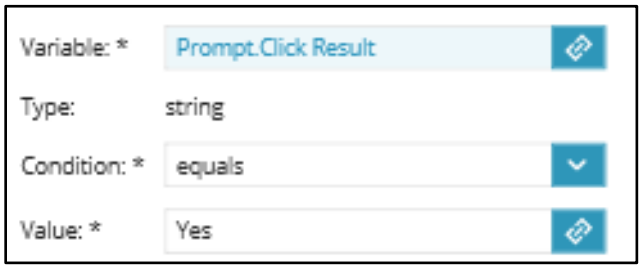
    b. In the left menu, search for the **If/Else** object and drag it to place it below the **Prompt** object.
    c. In the left menu, search for the **Delete Temporary Files** object and drag it into the **Then** section of the **If/Else** object.
    d. In the left menu, search for the **Log** object and drag it into the **Else** section of the **If/Else** object.
3. Enter the appropriate information in the objects' fields, and link required objects:
    a. In the **Prompt** object, fill the fields as indicated below:
        i. **Title**: Maintenance Engine
        ii. **Prompt**: Hi, can we do a quick maintenance check?
        iii. **Wait**: 600 seconds
        iv. **Buttons**: Yes/No
    b. In the **If/Else** object:

i. Click on the link icon ( )  beside the **Variable** field.

ii. In the **Object Link** pop-up, under **Objects** click **Prompt**, then under **Ouptput parameters**, click **Click Result**, and click **OK** to save.



iii. Select **Equals**, from the **Condition** drop-down list.
iv. The object will look like this:



c. In the **Log** object, enter a message noting that the user refused the maintenance.

d. Run the policy by clicking the play  button.
e. Save the file to your local drive for later use.

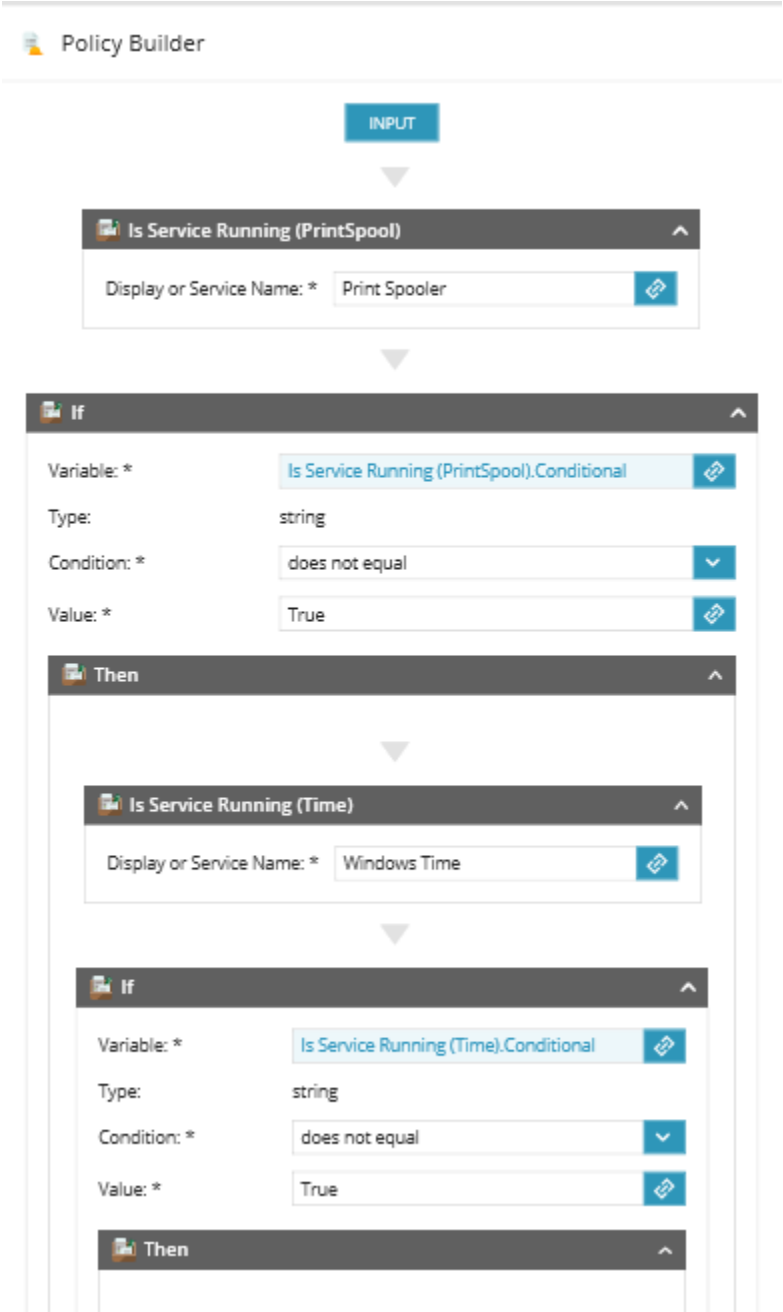
Optional steps

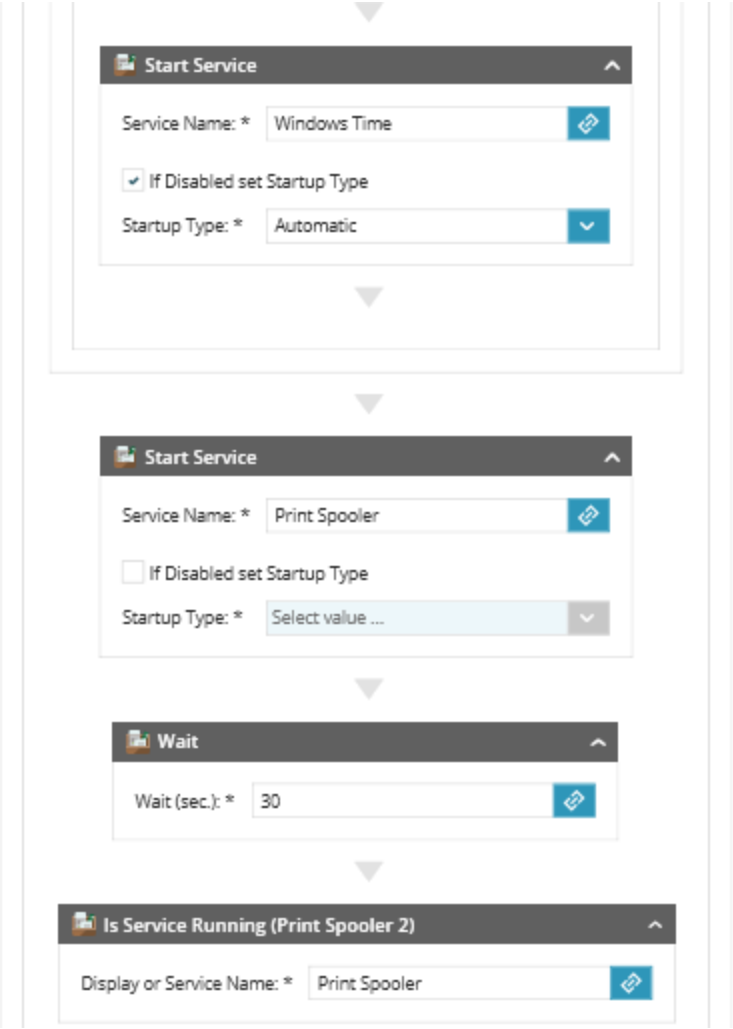
1. You can run the policy on your computer multiple times by using **Yes/No** as options. You can also try not answering at all. You may need to reduce the wait to 10 seconds to test the **No** answer.
2. You can try to upload the policy to your server and run it on your own computer.

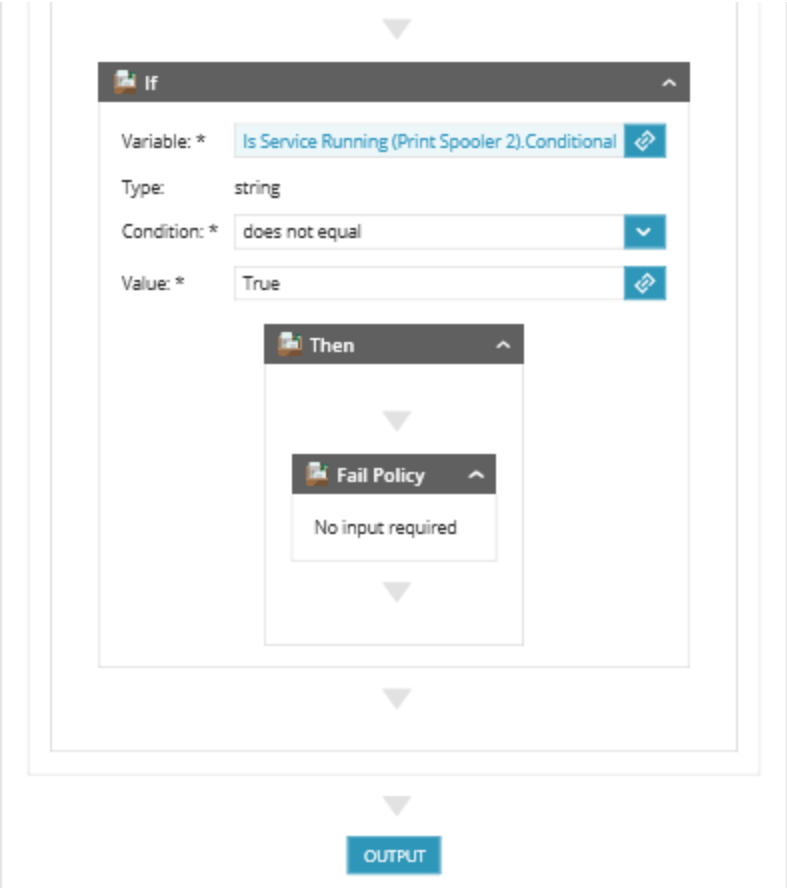
Notes:

Where could I use this?

Lab 2 – Proactive maintenance







Objective

Create a policy that can restart a windows service and look for a dependency service to ensure it is running. This is useful for line-of-business applications that have services with pre-requisites that must be started in sequence.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

For this lab, you will create an Automation Policy called **LAB02 – *<YOURNAME>***.

The goal is to check if the **Print Spooler** service is running. If it is not running, we need to check if the dependency service, **Windows Time**, is running. If it is running, we want to restart it, and if it is not running, we want to start it.

Then we need to start the **Print Spooler** service and after a five second wait time, we need to check if the service is running. If it is not running, we need to fail the policy execution to trigger a failed status.

Detailed steps

1. Create a new policy in automation manager
    a. Name it **LAB02 – *<YOURNAME>*** and describe it as "Lab 02 - Restart Windows Service".
2. Add the required objects (search for and drag the objects as you did in Lab 1):
    a. Add **Is Service Running** object below **Input** of the Policy Builder.
        i. Click the title bar for the object and add "(Print Spool)" to the object name.
        ii. In the **Display or Service Name** field, enter "Print Spooler".
    b. Add the **If** object:
        i. Click the link icon at the right of the **Variable** field.
        ii. In the **Object Link** pop-up, under **Objects** click **Is Service Running**.
        iii. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
        iv. In the **Condition** field, select **does not equal** from the drop-down list.
        v. In the **Value** field, enter ''True''. Note that this is case sensitive.
    c. Add the following objects to the **Then** section of the **If** object:
        i. Add the **Is service running** object:
            1. Click in the title bar and add ''(time)'' to the object name.
            2. In the **Display or Service Name** field, enter ''Windows Time".
        ii. Add the **If** object:
            1. Click the link icon at the right of the **Variable** field.
            2. In the **Object Link** pop-up, under **Objects** click **Is Service Running (time)**.
            3. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
            4. In the **Condition** field, select **does not equal** from the drop-down list.

5. In the **Value** field, enter ''True''. Note that this is case sensitive.
6. In the **Then** section of this new **If** object, add the **Start Service** object.
7. In the **Service Name** field, enter ''Windows Time".
8. Select the **If disabled set Startup Type** check box.
9. Select **Automatic** in the **Startup Type** dropdown list.

d. After the second **If/Then** object, but still within the first **Then** object (this may be confusing, refer to the screenshot above if required), add the following objects:
   i. Add the **Start Service** object.
      1. In the **Service Name** field, enter "Print spooler".
   ii. Add the **Wait** object.
      1. In the **Wait (sec)** field, enter ''30''.
   iii. Add the **Is service running** object.
      1. Click in the title bar and add "(Print Spool 2)" to the object name.
      2. In the **Display or Service Name** field, enter "Print Spooler".
   iv. Add another **If** object.
      1. Click the link icon at the right of the **Variable** field.
      2. In the **Object Link** pop-up, under **Objects** click **Is Service Running (print spool 2)**.
      3. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
      4. In the **Condition** field, select **does not equal** from the drop-down list.
      5. In the **Value** field, enter ''True''. Note that this is case sensitive.
      6. In the **Then** section of this new **If** object, add the **Fail Policy** object.

3. You can now run the policy and test it. Since we look at the Print Spooler and Windows Time services, this should not do anything as they should be running.
4. Save the file to your local drive for later use.

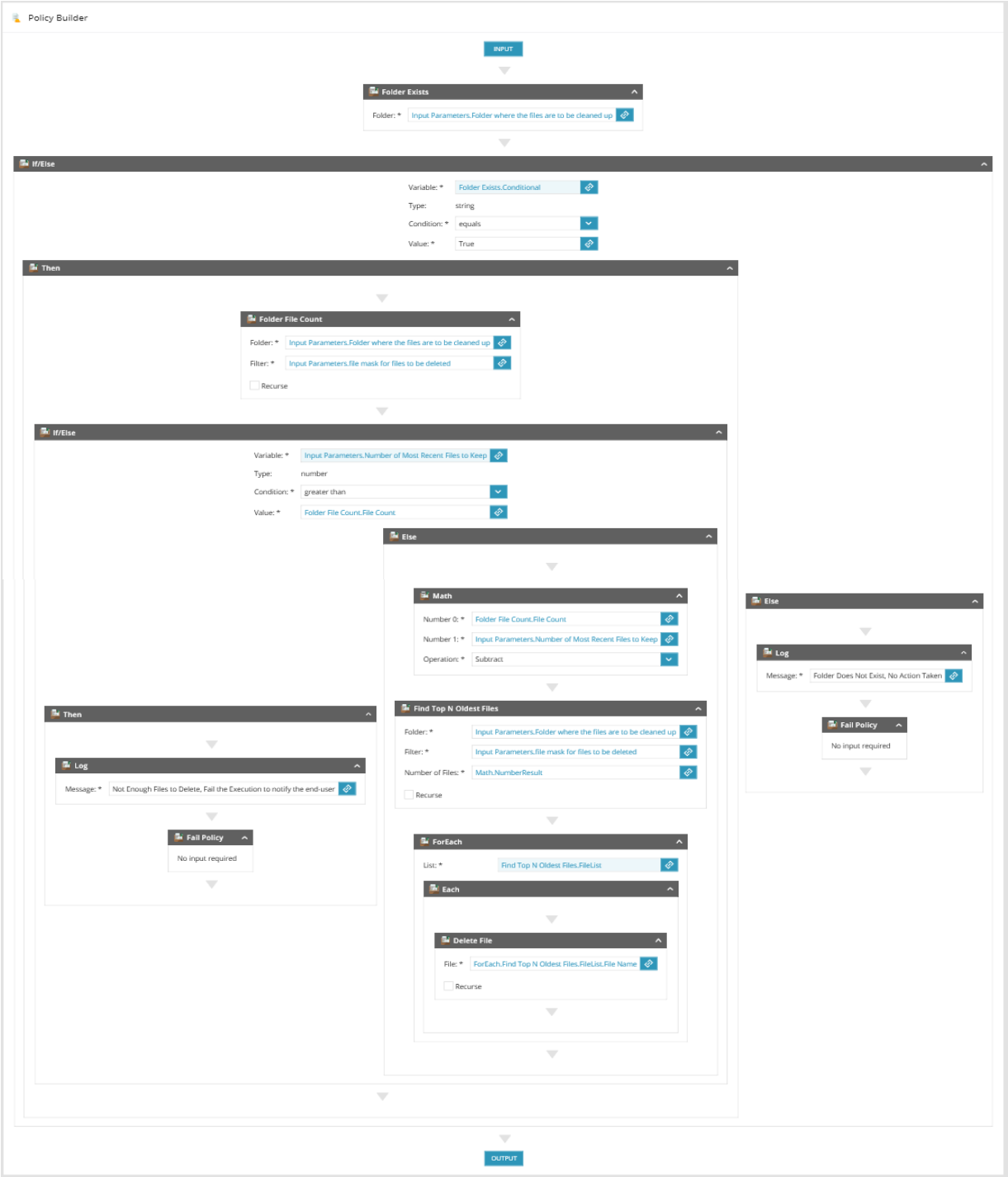
Optional steps

1. You can try to open the ''services.msc'' console and stop the Print Spooler service and run the policy again your computer. You will notice that the policy will run and restart the service.
2. You can try to upload it to your server and configure as self-healing on a Print Spooler Windows monitoring service (ask for help if you want to configure this during the course).

Notes:

Where could I use this?

Lab 3 – Proactive Maintenance



Objective

Create a policy that will look through a folder and keep a certain number of most recent files, ensuring that not all files are deleted. This is useful where backups stop running and you accidentally delete all files based on date, without noticing that new files are not being created.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

For this lab, you will create an automation policy called **LAB 03 - *<YOURNAME>***.

The goal is to have a policy that has three input parameters:

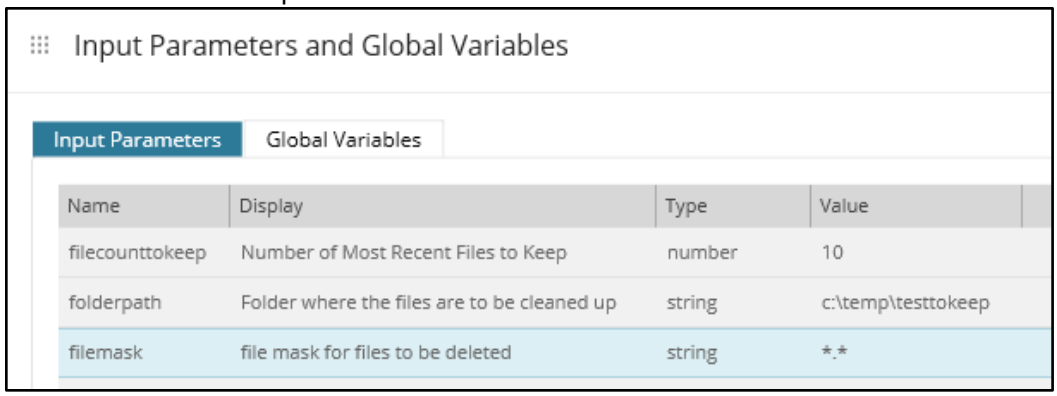
- A count of the number of files to keep in the folder.
- The path of the folder to check.
- The file mask (typically *.*).

The policy would need to check if the folder exists, and if it doesn't, a log object should be used to output a ''no update needed'' state. If the folder exists, it needs to get a file count, do a math calculation to get the number of files to delete (total files – files to keep), then feed that result to an object to find the count of the number of oldest files.

The policy would then need to loop through all the files found and delete them.

Detailed steps

1. Create a new policy in automation manager:
    a. Name it **LAB 03 - *<YOURNAME>*** and describe it as "Lab 03 Delete Old Files".
2. Click **INPUT** in the Policy Builder.
    a. Create three Input Parameters that match the information below:



Input Parameters and Global Variables

Input Parameters | Global Variables

Name	Display	Type	Value
filecounttokeep	Number of Most Recent Files to Keep	number	10
folderpath	Folder where the files are to be cleaned up	string	c:\temp\testtokeep
filemask	file mask for files to be deleted	string	*.*

3. Add the required objects (search for and drag the objects as you did in the previous labs).

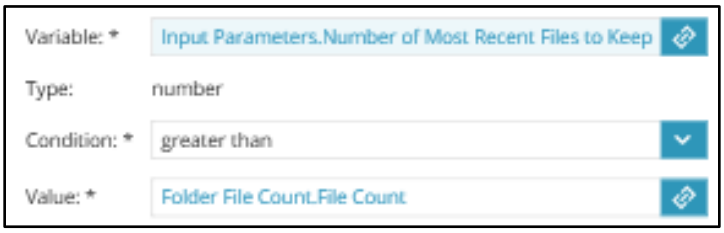
a. Add the **Folder Exists** object.
    i. Click the link icon at the right of the **Folder** field.
    ii. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
    iii. Under **Ouptput parameters**, click **Folder where the files are to be cleaned up**, and click **OK** to save.
b. Add an **If/Else** object.
    i. Click the link icon at the right of the **Variable** field.
    ii. In the **Object Link** pop-up, under **Objects** click **Folder Exists**.
    iii. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
    iv. In the **Condition** field, select **equals**.
    v. In the **Value** field, enter **True**.

    vi. The completed object will look like this:
c. In the **Else** section, add a **Log** object.
    i. In the **Message** field, enter ''Folder does not exist, no action taken.''
d. In the **Else** section immediately after the Log object, add a **Fail Policy** object.
e. In the **Then** section, add a **Folder File Count** object.
    i. Click the link icon at the right of the **Variable** field.
    ii. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
    iii. Under **Ouptput parameters**, click **Folder where the files are to be cleaned up**, and click **OK** to save.
    iv. Click the link icon at the right of the **Filter** field.

    v. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
    vi. Under **Ouptput parameters**, click **file mask for files to be deleted**, and click **OK** to save.
f. In the same **Then** section immediately below the **Folder File Count** object, add another

If/Else object.
    i. Click the link icon at the right of the **Variable** field.
    ii. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
    iii. Under **Ouptput parameters**, click **Number of Most Recent Files to Keep**, and click **OK** to save.
    iv. In the **Condition** field, select **greater than**.
    v. Click the link icon at the right of the **Value** field.
    vi. In the **Object Link** pop-up, under **Objects** click **Folder File Count**.
    vii. Under **Ouptput parameters**, click **File Count**, and click **OK** to save.

viii. The completed object will look like this:



g. In the **Then** section of this second **If/Else** object, add a **Log** object.
   i. In the Message field, enter ''Not enough files in the folder to require deletion. Failing policy.''
h. In the same **Then** section immediately below the **Log** object, add a **Fail Policy** object.
i. In the **Else** section of this second **If/Else** object, add a **Math** object.
   i. Click the link icon at the right of the **Number 0** field.
   ii. In the **Object Link** pop-up, under **Objects** click **Folder File Count**.
   iii. Under **Ouptput parameters**, click **File Count**, and click **OK** to save.
   iv. Click the link icon at the right of the **Number 1** field.
   v. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
   vi. Under **Ouptput parameters**, click **Number of Most Recent Files to Keep**, and click **OK** to save.
   vii. In the Operation field, select **Subtract**.
j. In the same **Else** section immediately below the **Math** object, add a **Find Top N Oldest Files** object.
   i. Click the link icon at the right of the **Folder** field.
   ii. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.
   iii. Under **Ouptput parameters**, click **Folder where the files are to be cleaned up**, and click **OK** to save.

   iv. Click the link icon at the right of the **Filter** field.
   v. In the **Object Link** pop-up, under **Objects** click **Input Parameters**.

   vi. Under **Ouptput parameters**, click **file mask for files to be deleted**, and click **OK** to save.
   vii. Click the link icon at the right of the **Number of Files** field.
   viii. In the **Object Link** pop-up, under **Objects** click **Math**.
   ix. Under **Ouptput parameters**, click **NumberResult**, and click **OK** to save.
k. In the same **Else** section immediately below the **Find Top N Oldest Files** object, add a **For Each** object.
   i. Click the link icon at the right of the **List** field.
   ii. In the **Object Link** pop-up, under **Objects** click **Find Top N Oldest Files**.
   iii. Under **Ouptput parameters**, click **FileList**, and click **OK** to save.
l. In the **Each** section of this **For Each** object, add a **Delete File** object.
   i. Click the link icon at the right of the **File** field.
   ii. In the **Object Link** pop-up, under **Objects** click **For Each**.
   iii. Under **Ouptput parameters**, click **FileName**, and click **OK** to save.

4. Save the file to your local drive for later use.

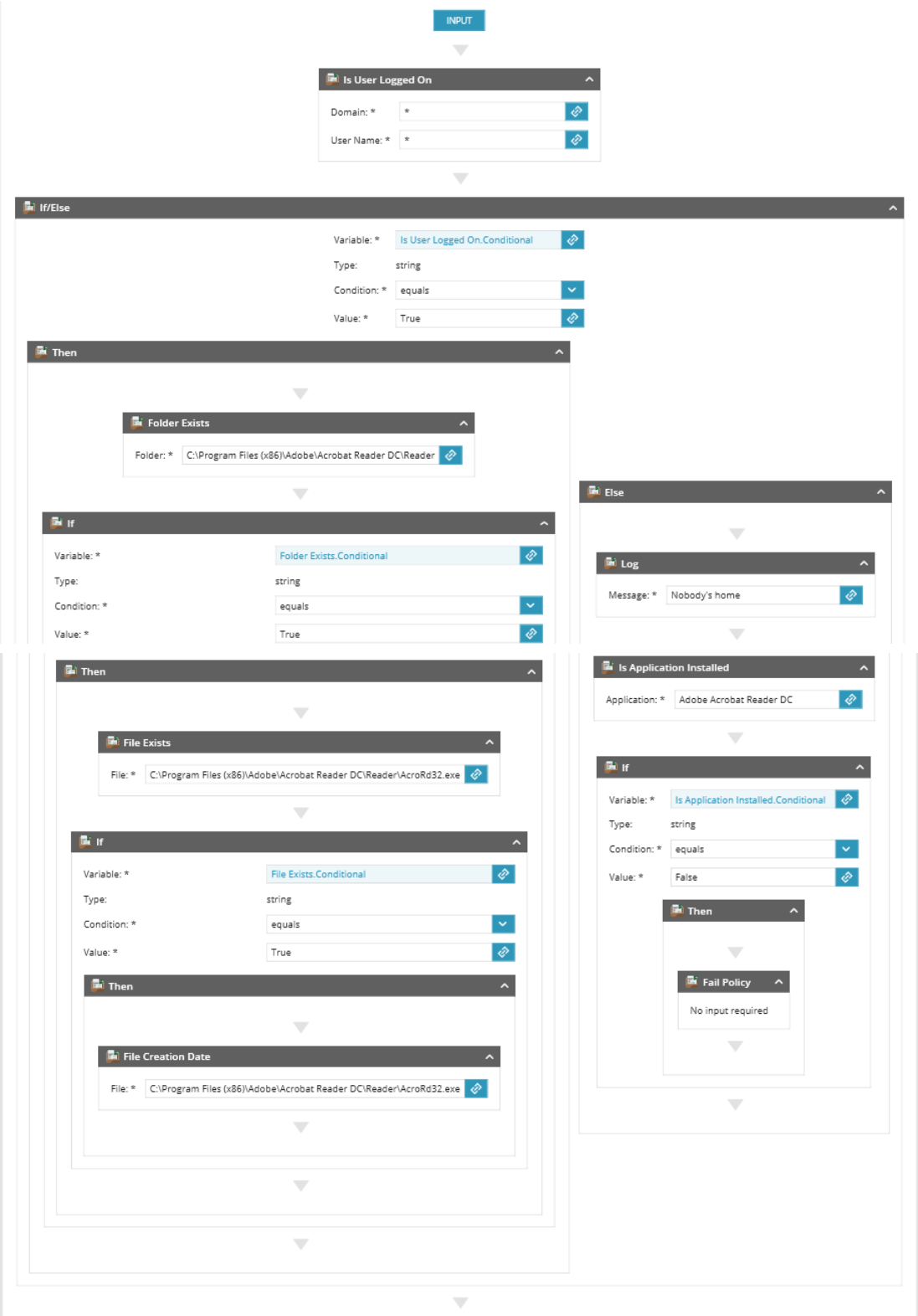
Optional steps

1. You can create a temp folder in C:\test and add 50 miscellaneous files, then run the policy on that folder and only keep 30. You will notice that the oldest files are removed.
2. You can try to upload it to your server and run it on your own computer to see its output. Ensure that the folder exists or try running the policy on a non-existent folder to see the result.

Notes:

Where could I use this?

Lab 4 – Common objects



Objective

Create a policy that highlights the usefulness of some of the more common objects, and show the output values and how to use them.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

For this lab, you will create an automation policy called **LAB 04 - *<YOURNAME>***.

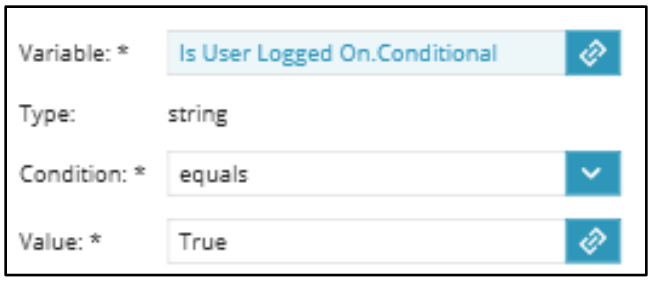
The end goal is to have a policy that follows this sequence:

1. Check if a user is logged on
    a. If a user is logged on, we need to check if the following folder exists: *C:\program files (x86)\adobe\acrobat reader dc\reader*.
        i. If the folder exists, check if the following file exists: *C:\program files (x86)\adobe\acrobat reader dc\reader\acrord32.exe*.
            1. If the file exists, get the file creation date.
        ii. If the folder doesn't exist, use a log to say "Nobody's home.", and then check if an application called *Adobe Acrobat Reader DC* is installed. If it is not installed, fail the policy.

Detailed steps

1. Create a new policy in automation manager
    a. Name it **LAB 04 - *<YOURNAME>*** and describe it as "Lab 04 - Common Objects".
2. Add the required objects (search for and drag the objects as you did in the previous labs).
    a. Add the **Is User Logged On** object.
        i. Enter an asterisk (*) in both the **Domain** and User **Name fields**.
    b. Add an **If/Else** object.
        i. Click the link icon at the right of the **Variable** field.
        ii. In the **Object Link** pop-up, under **Objects** click **Is User Logged On**.
        iii. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
        iv. In the **Condition** field, select **equals**.
        v. In the **Value** field, type **True**.

vi. The completed object will look like this:



c. In the **Then** section, add a **Folder Exists** object.

i. In the **Folder** field, enter "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader".

ii. In the same **Then** section immediately below the **Folder Exists** object, add an **If** object.

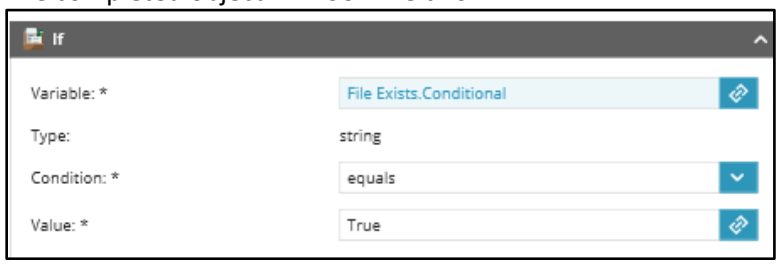
1. Click the link icon at the right of the **Variable** field.
2. In the **Object Link** pop-up, under **Objects** click **Folder Exists**.
3. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
4. In the **Condition** field, select **equals**.
5. In the **Value** field, type **True**.
6. The completed object will look like this:

iii. In the **Then** section of this same **If** object, add a **File Exists** object.

1. In the **File** field, enter "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe".

iv. In the same **Then** section immediately below the **File Exists** object, add another **If** object.

1. Click the link icon at the right of the **Variable** field.
2. In the **Object Link** pop-up, under **Objects** click **File Exists.**
3. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
4. In the **Condition** field, select **equals**.
5. In the **Value** field, type **True**.
6. The completed object will look like this:



v. In the **Then** section of this **If** object, add a **File Creation Date** object, and in the **File** field, enter: “C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe”.

d. Go back to the first **If** object and notice that the **Else** section is currently empty.

i. Add a **Log** object and in the **Message** field, enter “Nobody’s home”.

ii. Add an **Is Application Installed** object and in the **Application** field, enter “Adobe Acrobat Reader DC”.

iii. Add an **If** object.

1. Click the link icon at the right of the **Variable** field.
2. In the **Object Link** pop-up, under **Objects** click **Is Application Installed.**
3. Under **Ouptput parameters**, click **Conditional**, and click **OK** to save.
4. In the **Condition** field, select **equals**.
5. In the **Value** field, select **False**.

iv. In the **Then** section, add a **Fail Policy** object.

3. Save the file to your local drive for later use.

Optional steps

1. This simple policy shows you some of the common objects. You can try to modify it to check for various folders and files on your computer and observe if the policy finds those folders and files.
2. You can try to upload it to your server and run it on your own computer.

Notes:

Where could I use this?

Lab 5 – Uploading and maintaining a policy

Objective

This lab goes in RMM and shows how to interact with automation policies. This is useful to understand how to upload/download amps, and how to run them.

Estimated time

10 minutes

Required resources

The automation manager application

The Lab 01 automation policy file

Self-guided summary (advanced version)

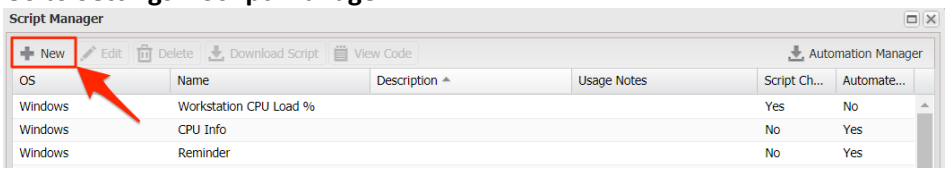
For this lab, you need to start with the LAB01 file.

Here's what you will need to do:

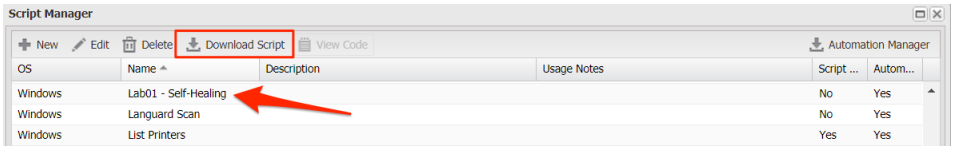
1. Upload the policy to the RMM dashboard.
2. View it in RMM under the automated tasks repository.
3. Finally, from RMM, download the AMP file.

Steps

1. Open the RMM dashboard in your browser.
2. Go to **Settings > Script Manager**.



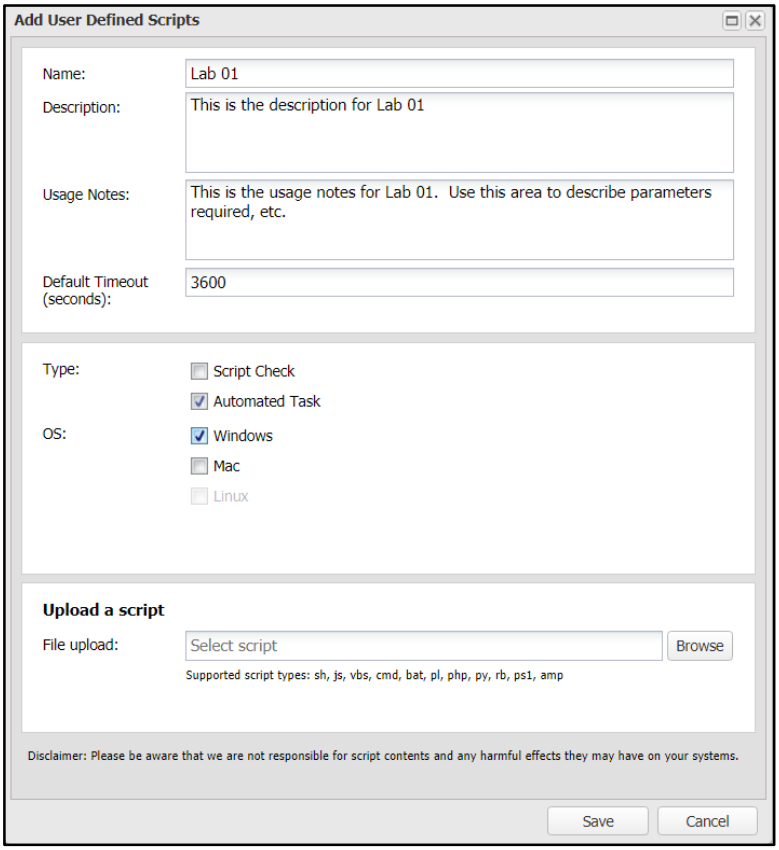
3. Click **New** in the upper-left corner of the Script Manager.
4. Enter the information for the task:
    a. **Name** (required)
    b. **Description** – What the task does.
    c. **Usage Notes** – How and when to use the task, what parameters need to be passed to it.
    d. **Default Timeout**



e. **Type** – Select the following:
   i. **Script Check** – Used for monitoring checks.
   ii. **Automated Task** – Used for maintenance and remediation type tasks.

f. **OS** – The operating system for which the task is designed.

g. **File Upload** – Browse to the AMP file that will be uploaded to your local device.



h. Click **Save** to start the upload.

5. In the Script Manager scroll down to the bottom to find your uploaded task.
6. Download files:
   a. In RMM, from the Script Manager, find your LAB 01 file. You can click the **Name** column to sort alphabetically.
   b. Select your LAB 01 file and click **Download** from the toolbar.

Important Notes

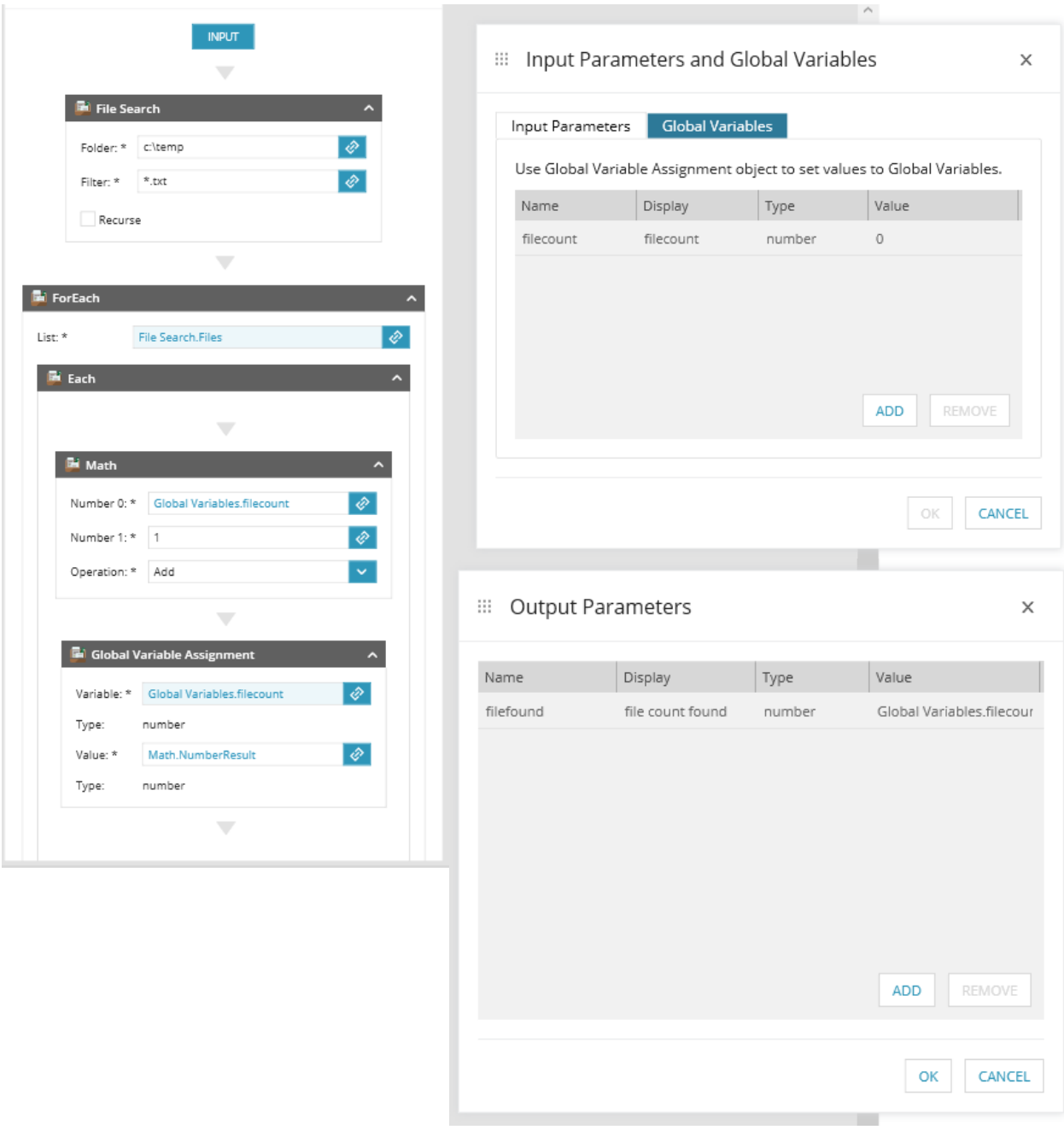
- When uploading an automation policy from outside of your organization, it is highly recommended to open it in automation manager first and review what it does, for security reasons.
- Also, when opening an automation policy in the you may get an error about the version. This is normal if the policy was created on a more recent version of automation manager.

Notes:

Where could I use this?

Lab 6 – Variables

Lab 6.1 – Numbers



Objective

To review how to interact with numbers and use math objects to do calculations.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

For this lab, you will need to create an automation policy called **LAB 06.1 – *<YOURNAME>*.**

The goal is to have a policy that looks in a folder, counts the number of files based on search criteria (using global variables), and outputs the count to an output parameter.

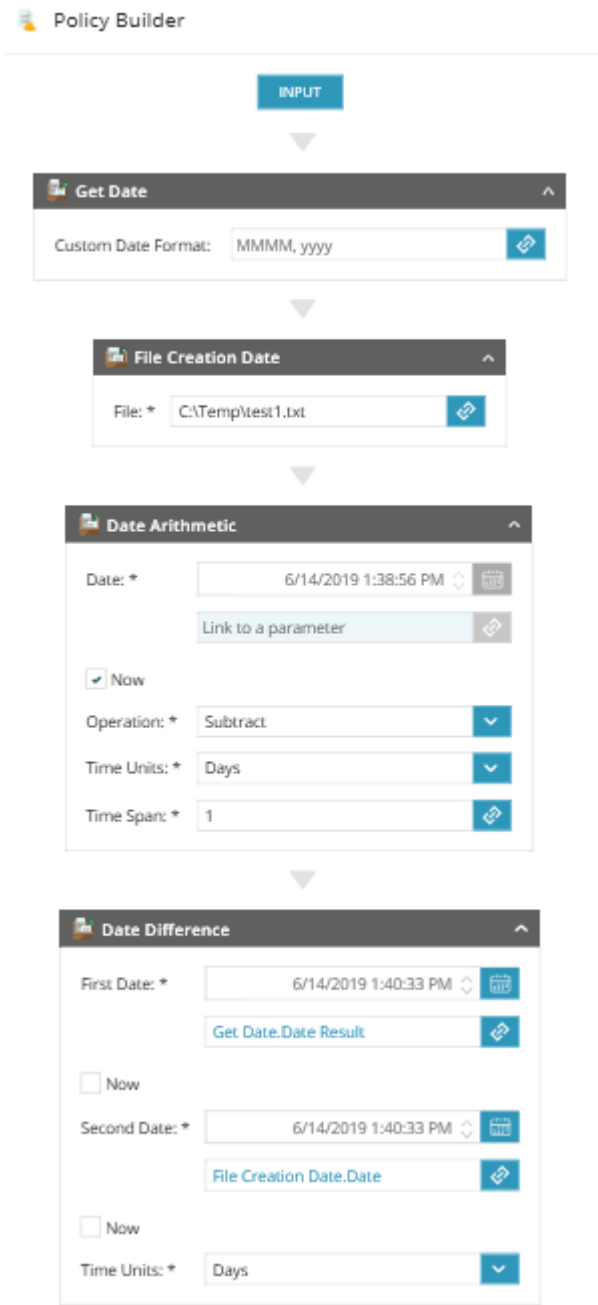
Steps

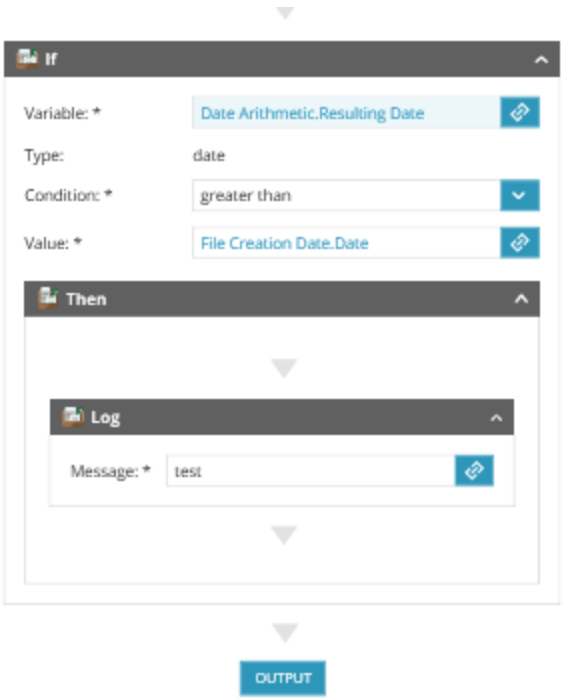
1. Create a new policy in automation manager.
    a. Name it **Lab 06.1 – numbers** and describe it as "Lab 06.1".
2. Create a global variable
    a. Click **INPUT** in the Policy Builder.
    b. Click **Global Variables**.
    c. Click **ADD**.
    d. Enter the following:
        i. In the **Name** field: "filecount".
        ii. In the **Display** field: "filecount".
        iii. In the **Type** dropdown list, select **number**.
        iv. In the **Value** field: 0.
    e. Click **OK** to save and **OK** again.
3. Add the required objects (search for and drag the objects as you did in the previous labs).
    a. Add a **File Search** object.
        i. In the **Folder** field, enter "C:\temp".
        ii. In the Filter field, enter "*.txt".
    b. Add a **For Each** object.
        i. Click the link icon at the right of the **List** field.
        ii. In the **Object Link** pop-up, under **Objects** click **File Search**.
        iii. Under **Ouptput parameters**, click **Files**, and click **OK** to save

    c. In the **Each** section, add a **Math** object.
        i. Click the link icon at the right of the **Number 0** field.
        ii. In the **Object Link** pop-up, under **Objects** click **Global Variables**.
        iii. Under **Ouptput parameters**, click **filecount**, and click **OK** to save.
        iv. and click **OK** to save.

        v. In the **Number 1** field, enter 1.
        vi. Select **Add** from the **Operation** drop-down list.
    d. In the same **Each** section immediately below the **Math** object, add a **Global Variable Assignment** object.
        i. Click the link icon at the right of the **Variable** field.
        ii. In the **Object Link** pop-up, under **Output Parameters** click **filecount**.
        iii. Click the link icon at the right of the **Value** field.
        iv. In the **Object Link** pop-up, under **Objects** click **Math**.
        v. Under **Ouptput parameters**, click **NumberResult**, and click **OK** to save.
4. Create an output parameter.
    a. Click **OUTPUT** in the Policy Builder.
    b. Click **ADD**.
    c. Enter the following:
        i. In the **Name** field: “ofilecount”.
        ii. In the **Display** field: “Total Files Found In Folder”.
        iii. Select **Number** from the Type drop-down list.
        iv. Link the **Value** field to the **filecount** Global Variable.
    d. Click **OK** to save and **OK** again to close.
5. Save the policy to your computer.

Lab 6.2 – Dates





Objective

This is meant to review the usage of date variables and how to do calculations on them

Estimated time

10 minutes

Required resources

Automation manager application

A text file created in C:\temp\test1.txt, or another file that can be used to compare dates.

Self-guided summary (advanced version)

For this lab, you will need to create an automation policy called **LAB 06.2 - *<YOURNAME>***.

The goal is to create a policy that retrieves a file creation date (C:\temp\file1.txt), then compares it to see if it is more than one day old.

Note that there are two objects that will do that calculation. Try to use both.

Steps

1. Create a new policy in automation manager
    a. Name it **LAB 06.2 - *<YOURNAME>***. and describe it as "Lab 06.2 - Date".
    b. Add the required objects (search for and drag the objects as you did in the previous labs).
        i. Add a **Get Date** object.
        ii. Add a **File Creation Date** object.
            1. In the **File** field, enter "C:\temp\test1.txt" or any file you put here to get a valid date.
        iii. Add a **Date Arithmetic** object.
            1. Select the **Now** check box.
            2. Select **Subtract** from the **Operation** drop-down list.
            3. Select **Days** from the **Time Units** drop-down list.
            4. In the **Time Span** field, enter "1".
        iv. Add a **Date Difference** object.
            1. For the **First Date**, link to the **Get Date** object and the **Date Result** Output Parameter.
            2. For the **Second Date**, link to the **File Creation Date** object and the **Date** Output Parameter.
            3. Select **Days** from the **Time Units** drop-down list.
        v. Add an **If** object.
            1. Link the **Variable** to the **Date Arithmetic** object and the **Resulting Date** Output Parameter.
            2. Select **greater than** from the **Condition** drop-down list.
            3. Link the **Value** to the **File Creation Date** Object and the **Date** Output

Parameter.

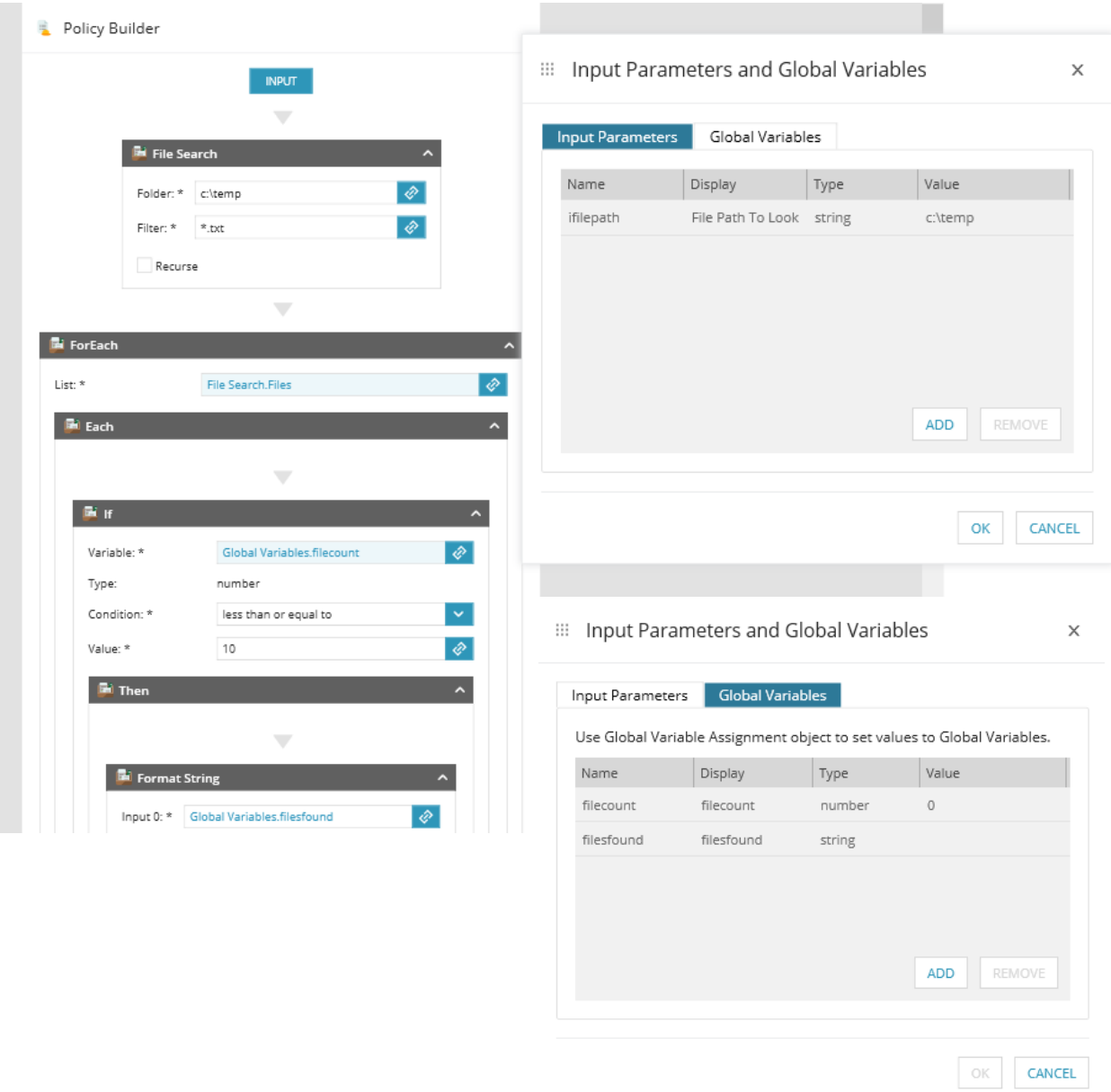
vi. In the **Then** section, add a **Log** object and enter “test” in the **Message** field.

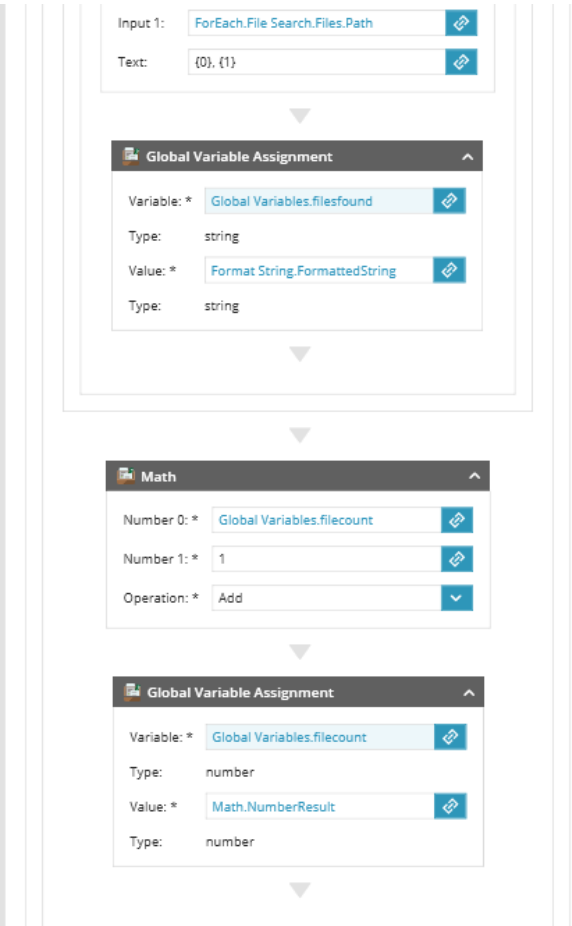
2. Save the policy to your computer.

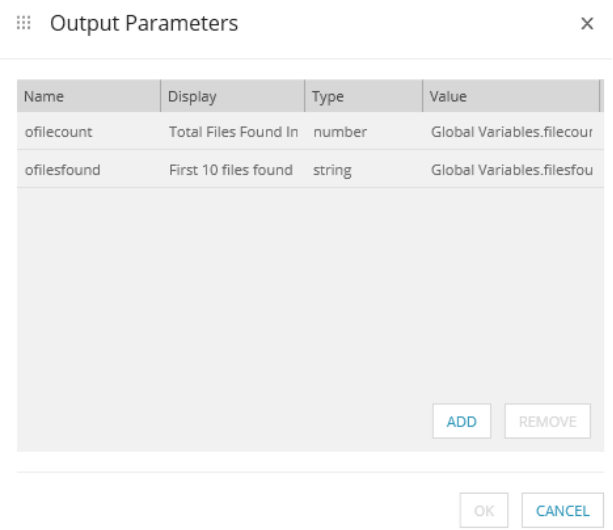
Notes:

Where could I use this?

Lab 7 – Input / output variables







Objective

To better understand how to use input/output variables in automation manager and RMM.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

For this lab, you will need to start from the LAB 6.1 policy file.

The goal is to modify the LAB 6.1 policy to use a dynamic file path as input, and to use that in the folder object, and then leverage a second global variable to append the first ten file names found, to an output parameter.

Steps

1. Open the automation policy LAB 6.1.
2. Save it as **LAB 7 – *<YOURNAME>***.
3. Click **Options** then **Properties**. Change name and description to be LAB 7.
4. Add an Input Parameter.
    a. Click **INPUT** in the Policy Builder.
    b. Click **ADD**.
    c. Click the **Input Parameters** tab.
    d. Enter the following:
        i. In the **Name** field: “filepath”.
        ii. In the **Display** field: “file path to look into”.
        iii. In the **Type** drop-down list, select string.
        iv. In the **Value** field: “C:\temp”.
    e. Click **OK**.
5. Add a Global Variable.
    a. Click **INPUT** in the Policy Builder.
    b. Click the **Global Variables** tab.
    c. Click **ADD**.
    d. Enter the following:
        i. In the **Name** field: “filesfound”.
        ii. In the **Display** field: “filesfound”.
        iii. In the **Type** drop-down list, select **string**.
        iv. Leave the **Value** field empty.
    e. Click **OK** to save, and **OK** again.
6. Add the following objects:

a. In the **Each** section of the **For Each** object immediately before the **Math** object, add an **If** object.
    i. In the **If** object, link the **Variable** field to the **Global Variables** object and the **filecount** Output Parameter.
    ii. Select **less than or equal to** from the **Condition** drop-down list.
    iii. In the **Value** field, enter 10.
    This will go in the **Then** section only if it has found 10 or less files.
b. In the **Then** section, add a **Format String** object.
    i. Link the **Input 0** field to the **Global Variables** object and the **filesfound** Output Parameter.
    ii. Link the **Input 1** field to the **foreach** object and the **path** Output Parameter.
    iii. In the **Text** field, enter {0}, {1}. This will format the string and concatenate the two strings together separated by a comma.
c. Add a Global Variable Assignment object immediately below the Format String object.
    i. Link the **Variable** field to the **Global Variables** object and the **filesfound** Output Parameter.
    ii. Link the **Value** field to the **Format String** object and the **FormattedString** Output Parameter.

7. Add an Output Parameter.
    a. Click OUTPUT in the Policy Builder.
    b. Click **ADD**.
    c. Enter the following:
        i. In the **Name** field: “ofilesfound”.
        ii. In the **Display** field: “First 10 files found in the folder that matched the criteria”.
        iii. In the **Type** drop-down list, select **string**.
        iv. Link the Value field to the Global Variables object and the filesfound Output Parameter.
        v. Click **OK** to save and click **OK** twice more.
8. Run the policy locally.
9. Save the policy to your local drive.

Notes:

Where could I use this?

Lab 8 – Custom check

Objective

This lab is intended to review the process to create a custom check in RMM.

Estimated time

10 minutes

Required resources

Automation manager application

Self-guided summary (advanced version)

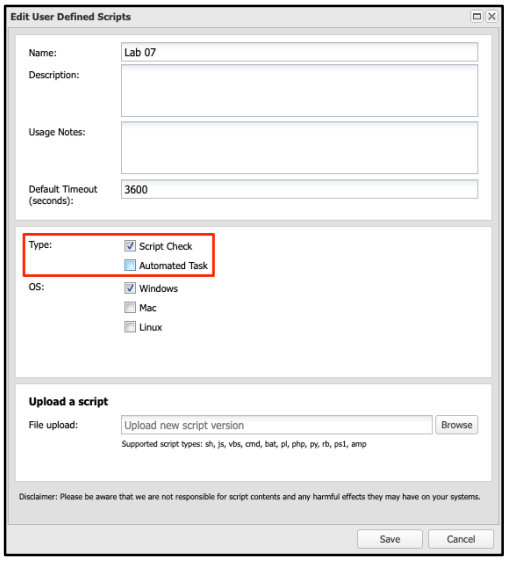
For this lab, you will need to start with the LAB 7 file.

The goal is to create a custom check from the LAB 7 file.

You will upload the policy to RMM, add the custom check to a device and run the check to observe the output.

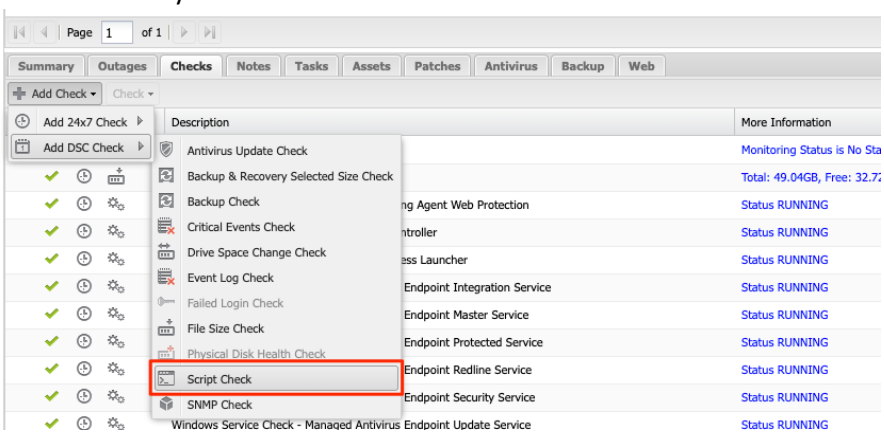
Steps

1. Upload and run the policy in RMM.

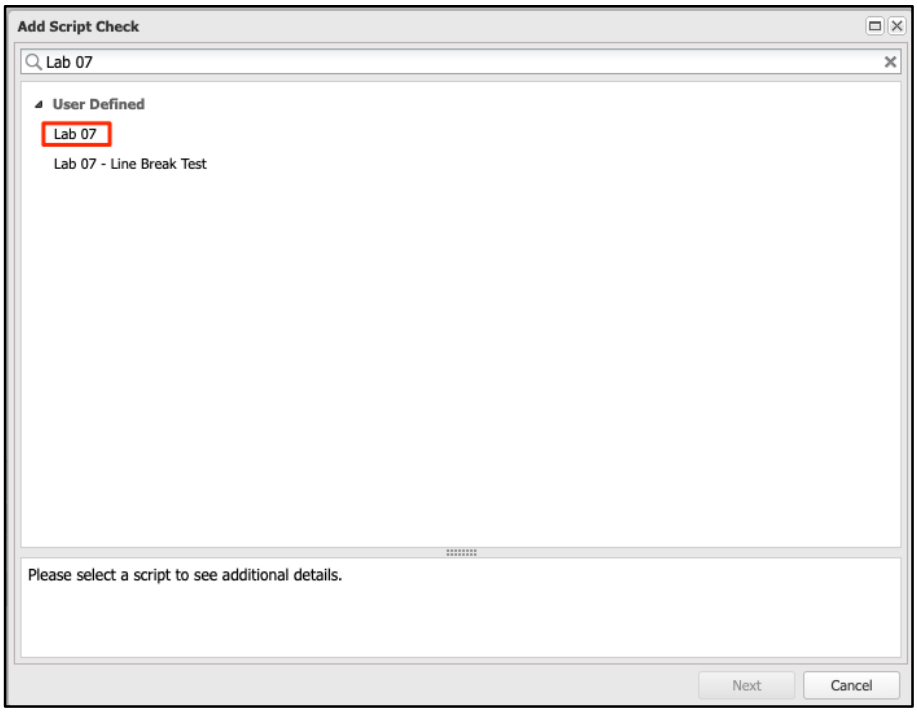


   a. Log in to RMM and upload the Lab 07 policy according to the steps in Lab #5.
      i. Select **Script Check** for the **Type**.

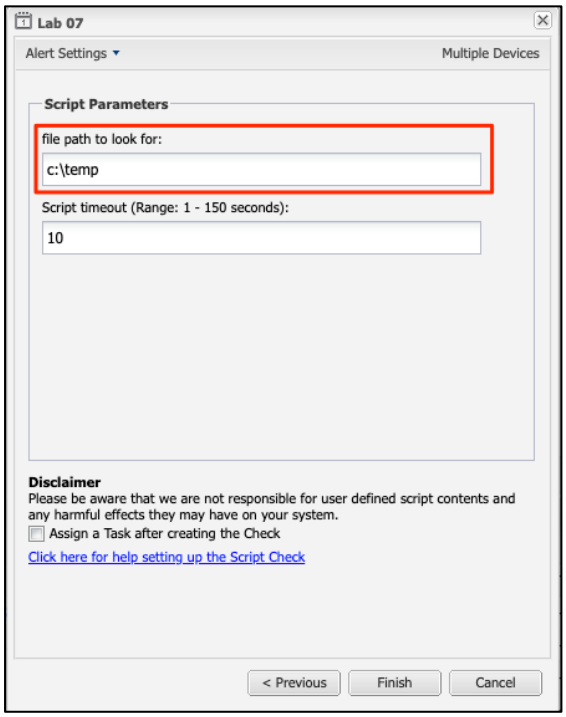
b. Select your lab device in the RMM dashboard and click the **Checks** tab.



c. Click **Add Check > Add DSC Check > Script Check**.
d. Select the Lab 07 task from the list (search or find it in the **User Defined** section).
e. Click **Next.**

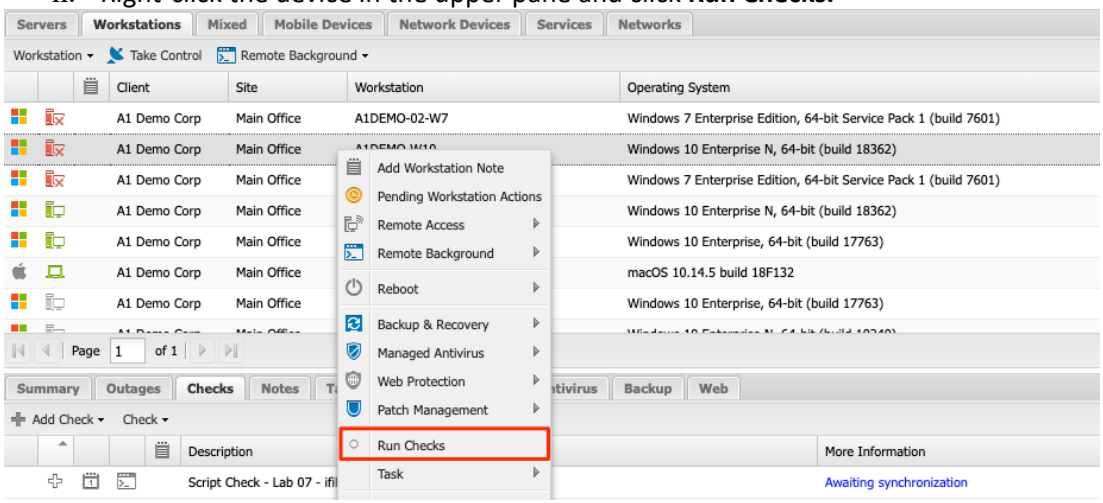


f. Enter the **file path to look for** parameter.



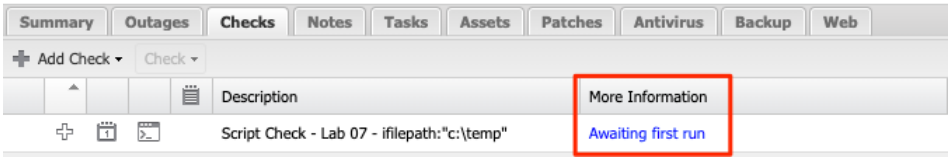
g. Click **Finish.**

h. Right-click the device in the upper pane and click **Run Checks.**



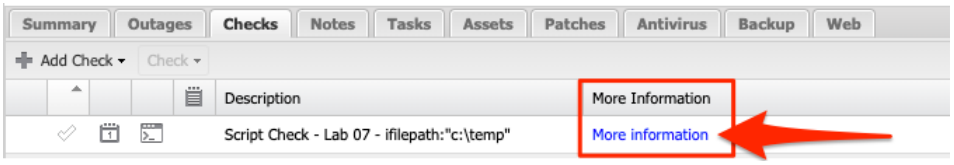
i. Wait 30 seconds and refresh the lower pane (Refresh icon in the upper-right corner of the lower pane).

j. When the **More Information** value for the Lab 07 check reads **Awaiting first run**, continue to step k.



k. Right click the Script Check – Lab 07 and click **Run Check**.

l. When the **More Information** value for the Lab 07 check reads **More information**, continue to step m.



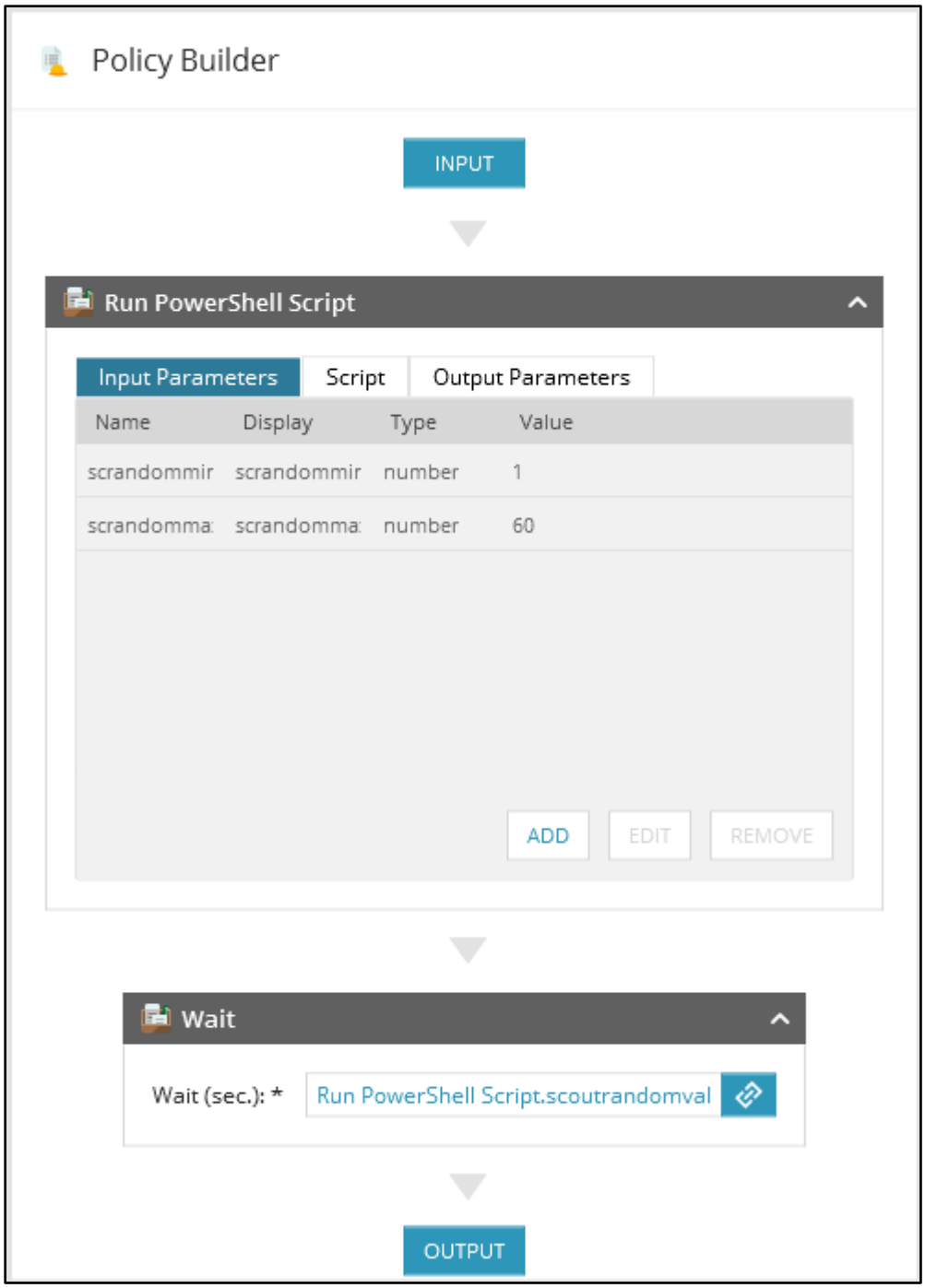
m. Click on the **More information** link for Script Check – Lab 07 to view the output.

n. The output variables will be used in the next two labs.

Notes:

Where could I use this?

Lab 9 – Using the run script object



Objective

To cover the basics of using your own script in a run script object with input/output variables.

Estimated time

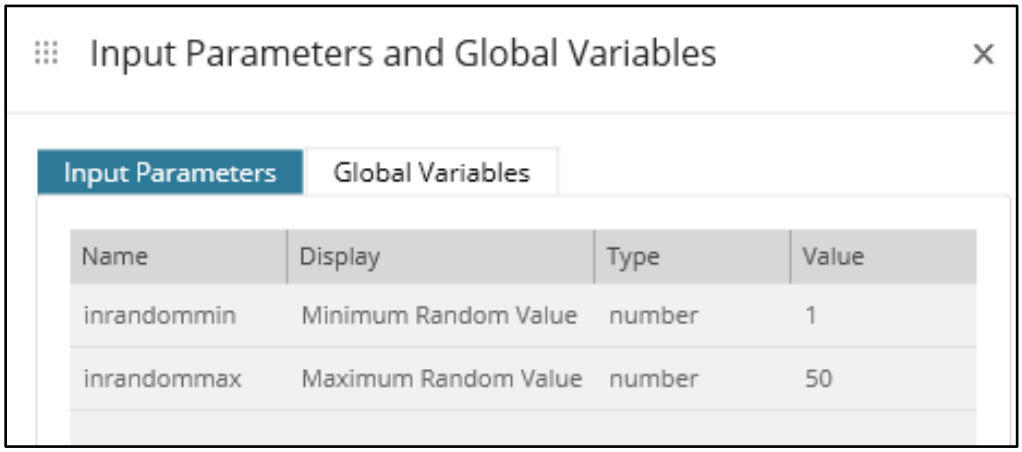
10 minutes

Required resources

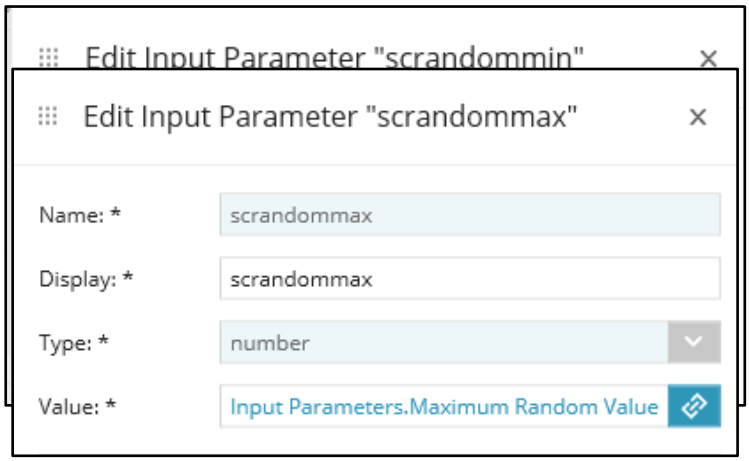
Automation manager application

Steps

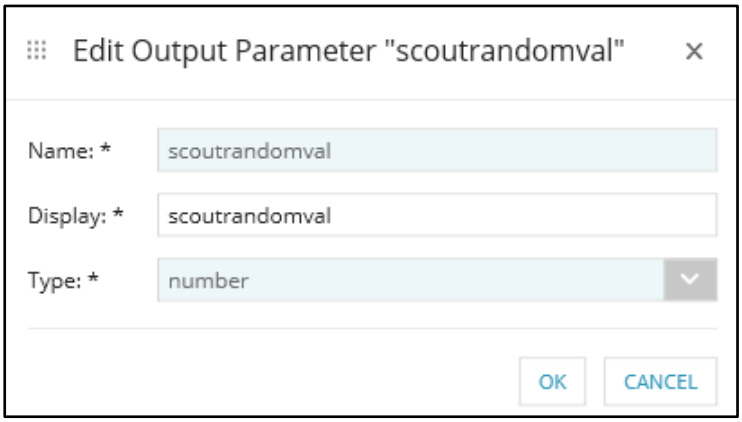
1. Create a new policy in automation manager
    a. Name it **Lab 10 – *<YOURNAME*>** and describe it as "Lab 10 - Run Script".
2. Add two Input Parameter as shown in the screenshot below:



3. Add the required objects (search for and drag the objects as you did in the previous labs).
    a. Add a **Run Powershell Script** object.
        i. Create 2 Input Parameters as shown in the screenshot below:



ii. Create one Output Parameter as shown in the screenshot below:



iii. Click the **Script** tab, click **EDIT** and Copy/paste the following script:
$randommin = $scrandommin
$randommax = $scrandommax
$irandomval = get-random - Minimum $randommin - Maximum $randommax
$scoutrandomval = $irandomval

4. Add a **Wait** object and link the **Wait (sec)** field to the output variable of the script.

Notes:

Where could I use this?