

E-Mail Automation Process using with Robusta RPA

In our process, we will first connect to an e-mail account and search for unread e-mails with a specific subject and an attached file. In the next step, we will forward each matching e-mail which meets certain condition to another e-mail account. For the e-mails which do not meet the condition, we will reply to the sender that the request is invalid.

- We started our process first by using the IMAP-SMTP Connection activity under the Mail section to connect to the e-mail account. Just drag and drop the activities you want to add to the process flow into the design area.

The screenshot displays the Robusta RPA interface. On the left, the 'Mail' section is expanded, showing various activities. The 'Imap/Smtp connection: Connect to e-mail account' activity is highlighted. The main design area shows a process flow starting with this connection activity, followed by a 'Script task: Find three days ago', a 'Search: Emails for flight routes price request', and a 'Get size: Request count' activity. A decision diamond follows, with a 'No request' path leading to an end node and a 'Request found' path leading to a 'Loop for each e-mail request'. Inside the loop, there is a 'Read/Save/Attachment: Download to processing folder' activity and a 'Get size: Attached file count' activity. The bottom panel shows the configuration for the 'Imap/Smtp connection: Connect to e-mail account' activity.

Imap/Smtp connection: Connect to e-mail account			
Id :	sid-521440C5-7655-4A...	Asynchronous :	<input checked="" type="checkbox"/>
Name :	Imap/Smtp connection ...	Documentation :	No value
Configuration name :	connectionName	* Connection name :	testCon
* Imap host :	imap.gmail.com	* Imap port :	993
* Imap user :	robustamailtest@gmail...	* Imap password :	Robustamail
Imap ssl :	<input checked="" type="checkbox"/>	Imap tls enabled :	<input type="checkbox"/>
Imap tls required :	<input type="checkbox"/>	* Smtp host :	smtp.gmail.com

Mail > Imap/Smtp connection	
Name	Imap/Smtp connection : Connect to E-mail account
Configuration Name	connectionName
*Connection name	testCon

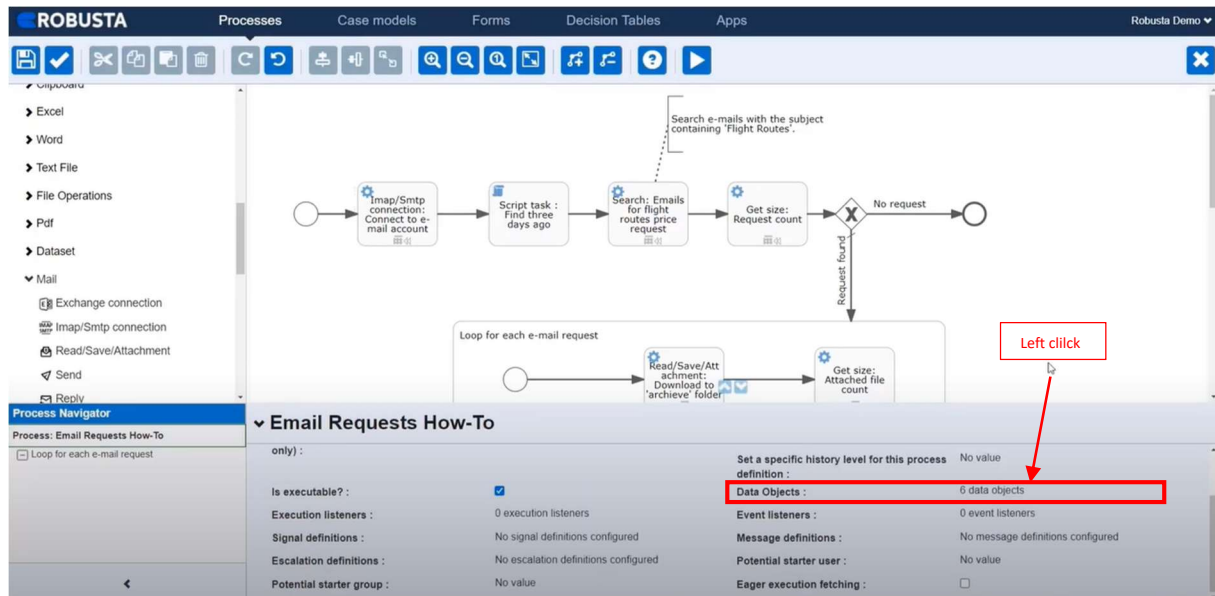
- The fields that need to be filled in order to connect to the Gmail account are given on the other page.
- While SMTP protocol is used for sending e-mail; IMAP protocol is used for e-mail reading. You can easily learn the parameter values for this component from the website of the e-mail account provider. Since we will connect to the Gmail account, we get this information and other necessary configurations from the relevant page of Google. <https://support.google.com/mail/answer/7126229?hl>
- This table contains information about how to set the parameters. According to the information here, we filled the relevant fields in the IMAP SMTP connection activity. We entered the user email that we want to connect to, in Imap user and Smtplib user fields.

Incoming Mail (IMAP) Server	imap.gmail.com
	Requires SSL: Yes
	Port: 993
Outgoing Mail (SMTP) Server	smtp.gmail.com
	Requires SSL: Yes
	Requires TLS: Yes (if available)
	Requires Authentication: Yes
	Port for SSL: 465
	Port for TLS/STARTTLS: 587
Full Name or Display Name	Your name
Account Name, User name, or Email address	Your full email address
Password	Your Gmail password

*Imap host	imap.gmail.com
*Imap user	E-mail address for IMTP protocol.
Imap ssl	<input checked="" type="checkbox"/>
*Smtplib port	587
*Smtplib password	\${password}
Smtplib tls enabled	<input checked="" type="checkbox"/>
*Imap port	993
*Imap password	\${password}
*Smtplib host	smtp.gmail.com
*Smtplib user	E-mail address for SMTP protocol.
Smtplib tls required	<input checked="" type="checkbox"/>

P.S: Enter the name of the user e-mail address you want to connect to.

- Then, in the Data Object field, which allows us to define a variable and use it in any activity in the process, we defined a variable named “password” and used it in the password fields.



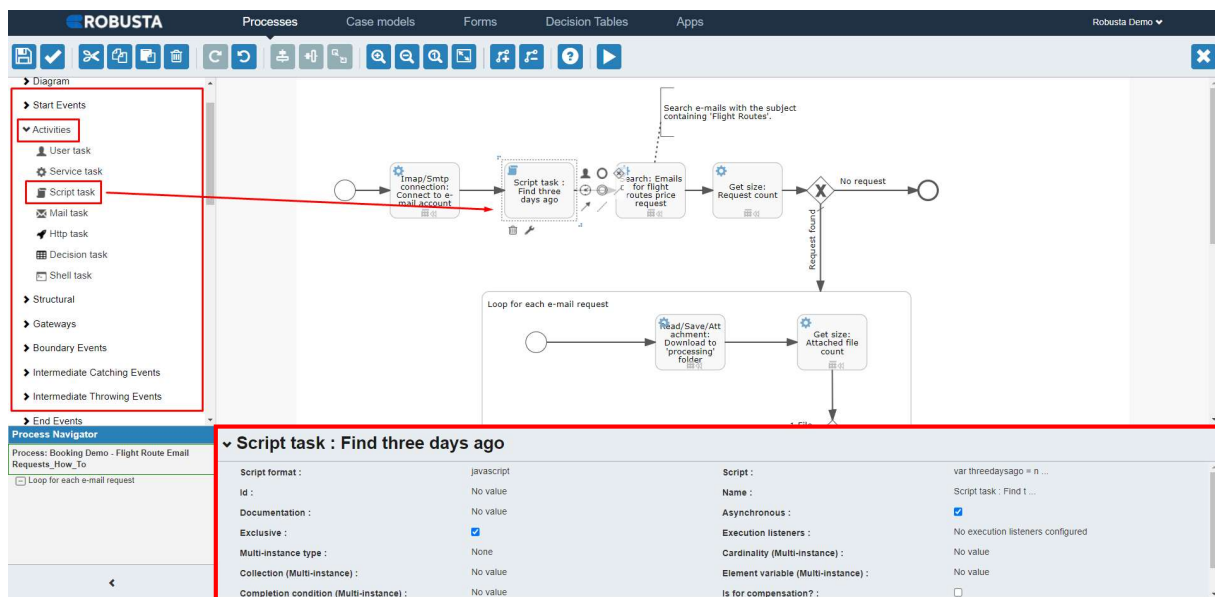
Change value for "Data Objects"

No data objects configured

Id	Name	Type	Default Value
minBpPriority	minBpPriority	string	51
humanEffort	humanEffort	string	0
outputPath	outputPath	string	C:\RobustaDemoProj...
successPath	successPath	string	C:\RobustaDemoProj...
failPath	failPath	string	C:\RobustaDemoProj...
password	password	string	

↑ ↓ + -

- After the connection is done, we found the date three days before today, which we want to use as the minimum date in Mail search activity. We used the javascript to find the date.
- The MinDate variable here will be used to find e-mails less than 3 days old. We wrote the variable name between curly braces after a dollar sign which is the standard way referring to a variable. Example: \${password}



Activities > Script task	
Script format	javascript
Name	Script task : Find three days ago
Script	<pre> var threeDaysAgo = new Date(); threeDaysAgo.setDate(threeDaysAgo.getDate()-3); var date = threeDaysAgo.toLocaleDateString(); var splitDate = date.split("-"); var year = splitDate[0]; var month = splitDate[1]; var day = splitDate[2]; var minDate = day+"."+month+"."+year; execution.setVariable("minDate", minDate); </pre>

- After this step, we used the Search activity under the Mail section and in the connection name field, we selected the connection reference name that we defined in the first activity from the list. Then, we found the e-mails with the subject containing the phrase “Flight Routes” and containing an attached file. For this, we wrote “Flight Routes” in the subject field and ticked the has attachment box. We did not change the default value of INBOX in the Folder Name field, as we want the relevant search to be in the Inbox. Then, by selecting the unread box, we searched for unread e-mails, and we defined a variable in the dataset field to assign the all e-mails’ data to a dataset.
- In this example, we set the e-mail limit to be searched as 10. This means that even if there are more than 10 e-mails matching the search criteria, only the 10 most recent e-mails will be imported into the dataset.

Search: Emails for flight routes price request

Id :	sid-204A1837-42A2-48 ...	Asynchronous :	<input checked="" type="checkbox"/>
Name :	Search: Emails for f ...	Documentation :	No value
* Connection name :	\${testCon}	* Folder name :	INBOX
From :	No value	Has attachment? :	<input checked="" type="checkbox"/>
Search limit :	10	Max date :	No value
Min date :	\${minDate}	Subject :	Flight Routes
Unread? :	<input checked="" type="checkbox"/>	* New Dataset name :	emailSearchDS

Mail > Search	
Name	Search : E-mails for flight routes price request
*Connection name	\${testCon}
Search limit	10
Min date	\${minDate}
Unread?	<input checked="" type="checkbox"/>
*Folder name	INBOX
Has attachment?	<input checked="" type="checkbox"/>
Subject	Flight Routes
*New Dataset name	emailSearchDS

- After the search activity, we found the number of e-mails transferred to the dataset and assigned the result to a variable. Since we want the rows of the dataset to be counted here, we chose the ROW option in the Size Type field from the list.

The screenshot shows the Robusta interface with a workflow diagram and the configuration for the 'Get size: Request count' dataset action.

Workflow Diagram:

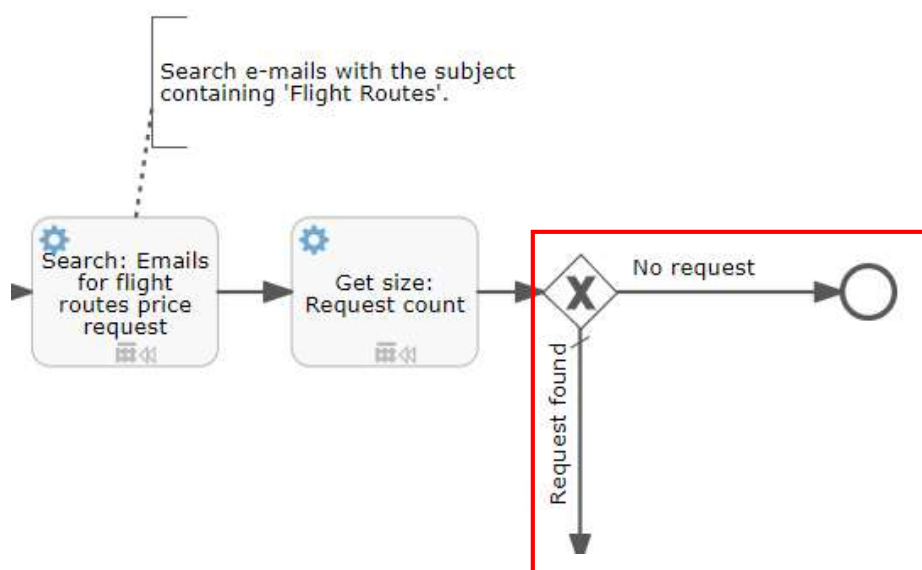
- Start node leads to 'Imap/Emtp connection: Connect to e-mail account'.
- Followed by 'Script task: Find three days ago'.
- Then 'Search: Emails for flight routes price request'.
- Then 'Get size: Request count' (highlighted with a red box).
- Decision node: 'Request four'.
- If 'No request', it goes to an end node.
- If 'Request four', it enters a 'Loop for each e-mail request'.
- Inside the loop, it goes to 'Read/Save/Attachment: Download to processing folder'.
- Then 'Get size: Attached file count'.
- End of loop.

Dataset Configuration (Get size: Request count):

- Name:** Get size: Request count
- * Dataset name:** \${emailSearchDS}
- * Result variable name:** emailSearchDSSize
- Asynchronous:** ☒
- Documentation:** No value
- Size Type:** ROW

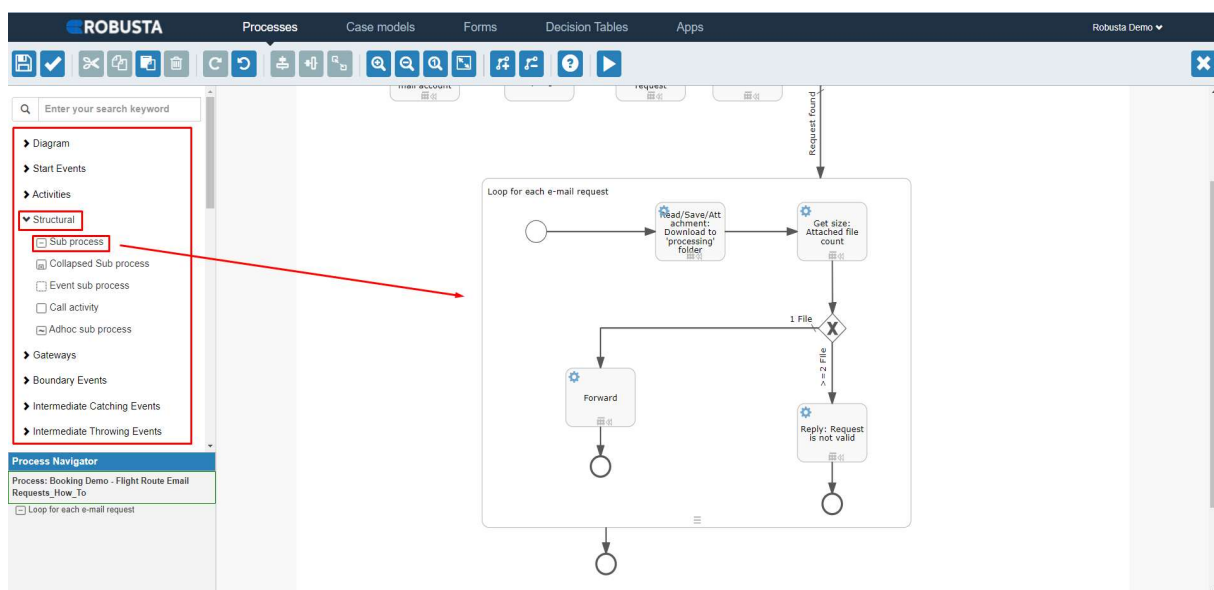
Dataset > Get size	
Name	Get size : Request count
*Dataset name	\${emailSearchDS}
*Result variable name	emailSearchDSSize
Size Type	ROW

- Then, the process is terminated if no e-mails is found, and the process continues if there is at least one e-mail. To do this, we used a gateway that allowed us to control how a process flows according to the conditions we set.
- For each arrow leaving the gateway we set a condition expression according to the e-mail count variable. If the value of this variable is 0, we ended the process flow. Otherwise, we chose the default flow option, and we did not set any condition expression. The process will continue from here if it does not match a condition.



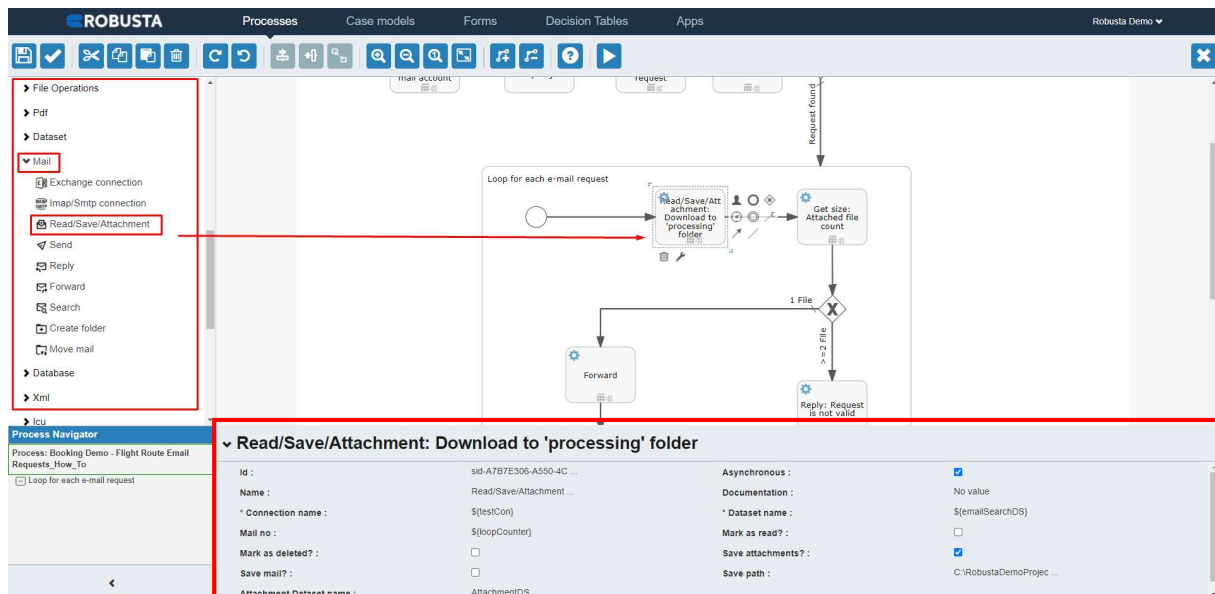
- “No request” flow condition ($\${emailSearchDSSize==0}$) checks whether emailSearchDSSize variable value is equal to ‘0’ which means there is no row in file. If this condition is met the process is completed with an end event. In the “Request found” flow condition, we chose the “default flow” option checked ☒ and did not set any condition expression.

- The process continues with a loop activity, since the operations will be done independently for each e-mail. We included the Sub-process activity from the Structure section to create a loop activity in the process. As in our main process, we need to create a flow with a beginning and end event in this sub-process.
- We need to define how many times this sub-process will be repeated in the Cardinality field. This will be provided by using emailSearchDSSize variable.
- We chose the value of the Multi-instance type field as Sequential because we want each operation to be done sequentially. When we make the definitions, the “loopCounter” variable is automatically defined at the beginning of the loop. This variable, which first takes a value of 0 increases by 1 at each iteration. When the value of the “loopCounter” variable reaches the Cardinality value, the loop is automatically terminated.



Structural > Sub process	
Name	Loop for each e-mail request
Cardinality	\${emailSearchDSSize}
Multi-instance type	Sequential
Exclusive	<input checked="" type="checkbox"/>

- In the loop, we first ensured that the e-mail we are processing with the Read/Save/Attachment activity is marked as read, the attached files list are transferred to a dataset, and the files attached are downloaded to the desired directory for archiving purposes.



Mail > Read/Save/Attachment	
Name	Read/Save/Attachment : Dowload to 'archieve' folder
*Connection name	\$(testCon)
Mail no	\$(loopCounter)
Attachment Dataset name	AttachmentDS
*Dataset name	emailSearchDS
Mark as read?	<input checked="" type="checkbox"/>
Save attachments?	<input checked="" type="checkbox"/>
Save path	The location you want to save the files.

- In this step, similar to the use of the Get Size activity that we explained in the first parts of the process, we have found the row number of the dataset to which the list of the attached files is transferred.

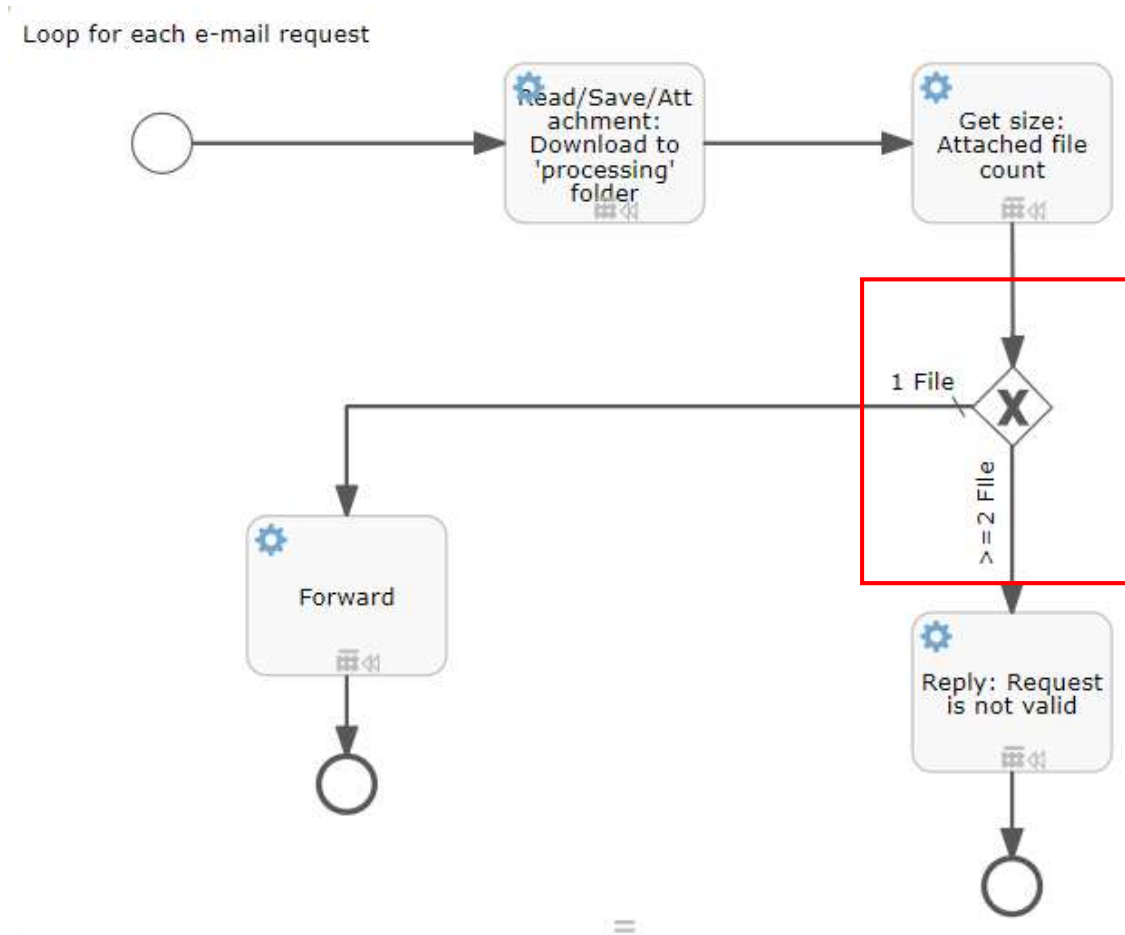
The screenshot displays the Robusta IDE interface. On the left, a sidebar contains a 'File Operations' menu with 'Dataset' expanded, and 'Get size' highlighted. A red line connects this menu item to the 'Get size: Attached file count' activity in the main workflow canvas. The workflow includes a 'Loop for each e-mail request' containing a 'Read/Save/Attachment: Download to processing folder' activity, followed by the 'Get size: Attached file count' activity. Below the canvas, a detailed configuration panel for the selected activity is shown.

Get size: Attached file count

Id :	sid-D7E8E636-4DEF-4D ...	Asynchronous :	<input checked="" type="checkbox"/>
Name :	Get size: Attached f...	Documentation :	No value
* Dataset name :	\${AttachmentDS}	Size Type :	ROW
* Result variable name :	listFileDSSize		

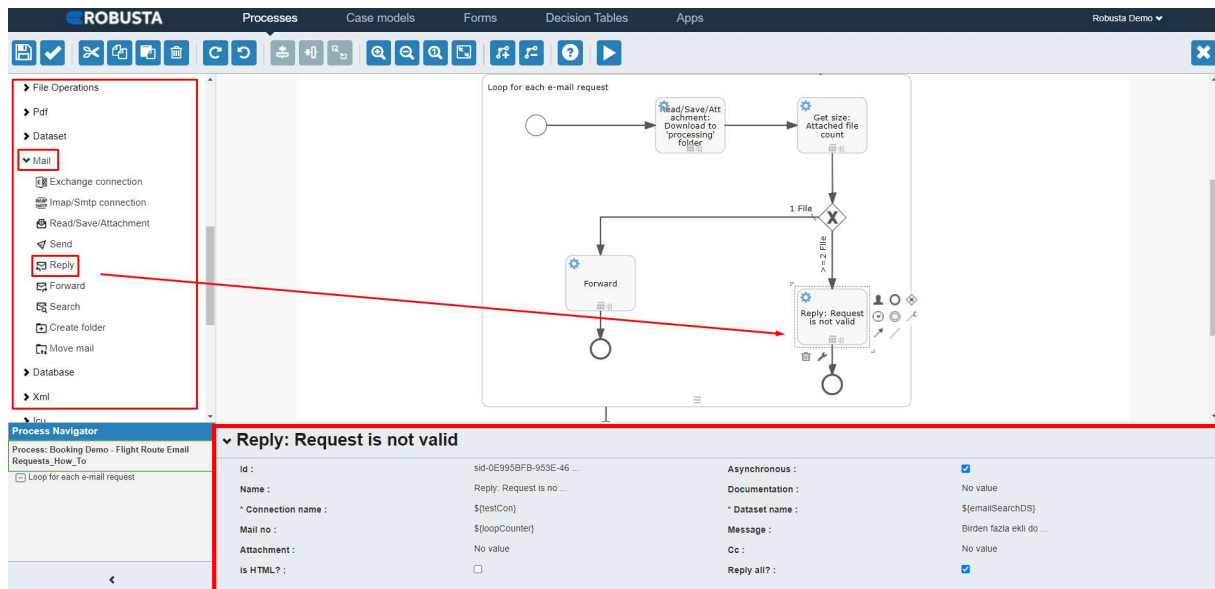
Dataset > Get size	
Name	Get size : Attached file count
*Dataset name	\${attachmentDS}
*Result variable name	listFileDSSize
Size Type	ROW

- After finding the number of attached files, we added another gateway to the process as we want to perform a different operation if there is more than 1 attached file, and a different operation if there is only 1 file.



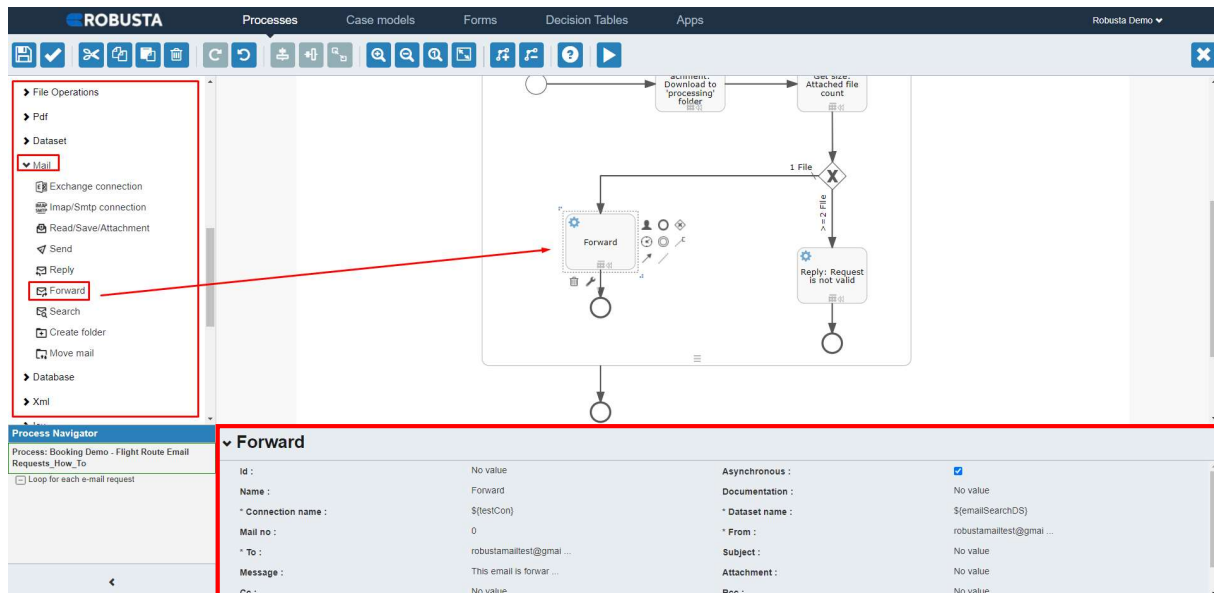
- “>=2 File” flow condition ($\${attachmentDSSize} \geq 2$) checks whether attachmentDSSize variable value is greater than or equal to “2” which means there are more than one attached file. In the “1 File” flow condition, we chose the “default flow” option ☒ checked and did not set any condition expression.

- If there are more than one attached file, the flow continues with Reply activity, and the message “Cannot process more than one attached file” is returned as a response to the sender. Then we completed our Sub process flow with an End event.



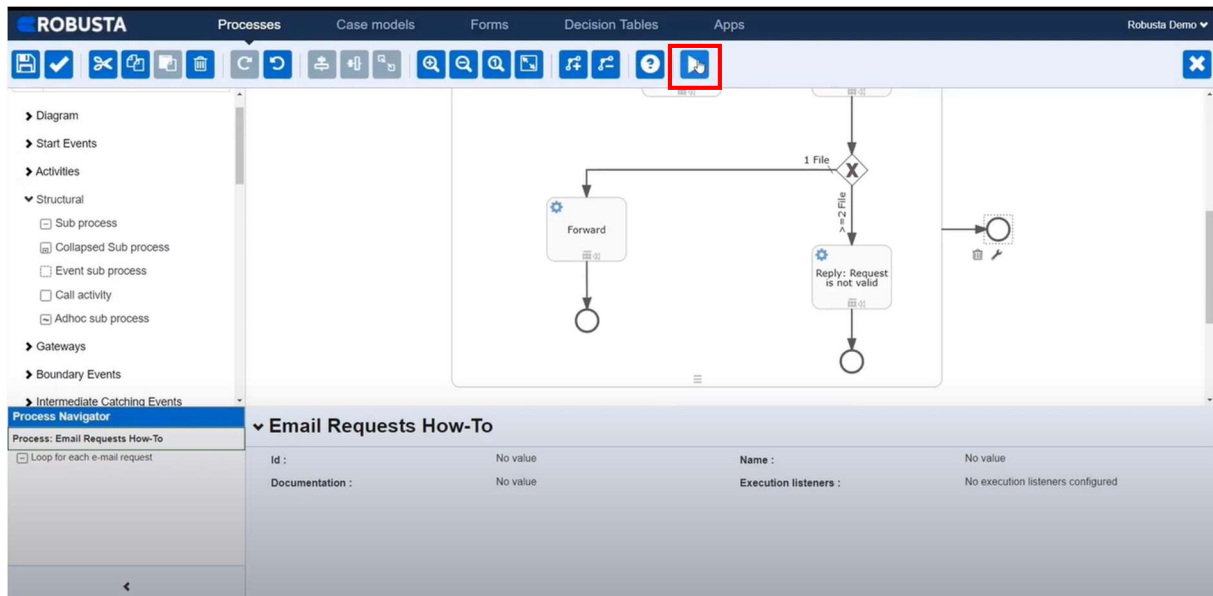
Mail > Reply	
Name	Reply : Request is not valid
*Connection name	\${testCon}
Mail no	\${loopCounter}
Is HTML?	<input checked="" type="checkbox"/>
*Dataset name	\${emailSearchDS}
Message	Cannot process more than one attached file
Reply all?	<input checked="" type="checkbox"/>

- If there is 1 file attached, the e-mail is forwarded to another e-mail address with the Forward activity. In this activity, the from and to e-mail addresses, e-mail no, subject, and message fields are defined. We completed our subprocess flow with an End event. Then we used another end event to terminate the main process.

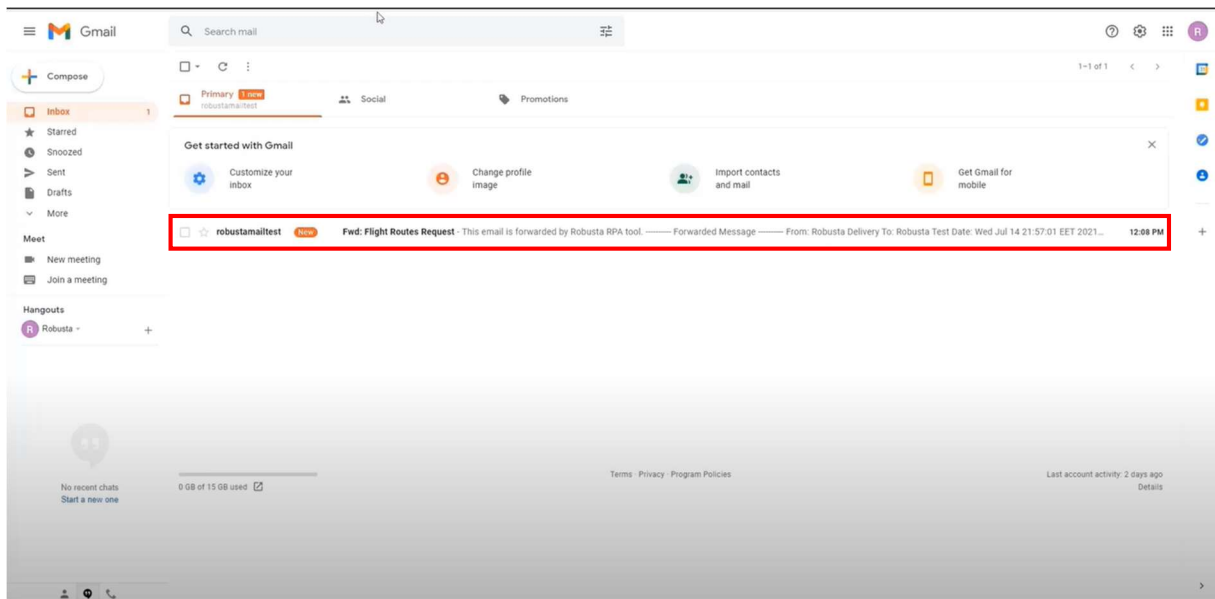
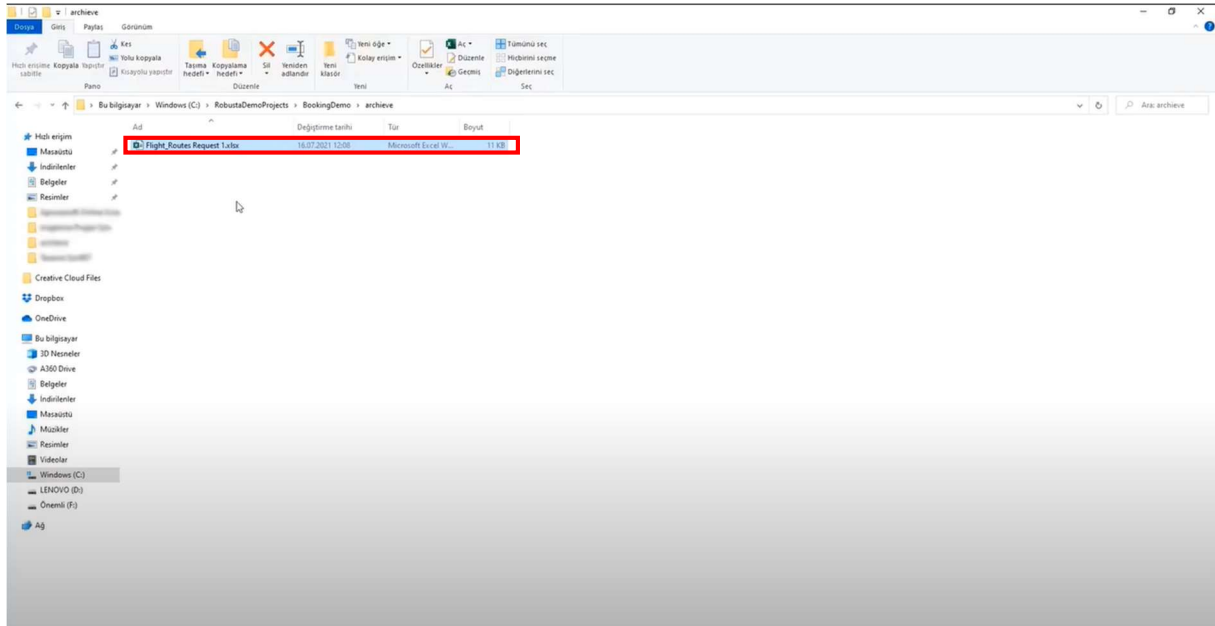


Mail > Forward	
Name	Forward
*Connection name	\${testCon}
Mail no	\${loopCounter}
*To	recipient's e-mail address
Message	This mail is forwarded by Robusta RPA tool.
*Dataset name	\${emailSearchDS}
*From	sender's e-mail address

- Now let's run our process by clicking the Run button. Since the activities used in this process work at the backend, we do not see any action on the screen.



- As a result, the attachments are downloaded to the specified folder and the e-mail is successfully forwarded to our e-mail account.



We came to the end of our How-to. We hope it was useful for you.