# 

Creating Workflow

**Lab Guide**



# Table of Contents

[1](#_Toc54007066)

[Table of Contents 3](#_Toc54007067)

[Document History 4](#_Toc54007068)

[Module 2: Creating a Quantum Fabric Workflow 4](#_Toc54007069)

[Goal 4](#_Toc54007070)

[Tasks 4](#_Toc54007071)

[Required Software and Tools 5](#_Toc54007072)

[**Required:** 5](#_Toc54007073)

[**Optional:** 5](#_Toc54007074)

[Module 2: Lab Instructions 5](#_Toc54007075)

[Step 1: Import the Fabric App 5](#_Toc54007076)

[Step 2: Create a Workflow 8](#_Toc54007077)

[Step 3: Add a Task to Submit an Application 10](#_Toc54007078)

[Step 4: Get the Loan Applicant's Score 11](#_Toc54007079)

[Step 5: Update the Loan Application 14](#_Toc54007080)

[Step 6: Test Our Progress So Far (Optional) 16](#_Toc54007081)

[Step 7: Assess the Score and Update the Loan Application 20](#_Toc54007082)

[Step 8: Test Our Progress So Far (Optional) 23](#_Toc54007083)

[Step 9: Send an Approval Notification 23](#_Toc54007084)

[Step 10: Send a Rejection Notification 25](#_Toc54007085)

[Step 11: Make Room for a Human to Decide 27](#_Toc54007086)

[Step 12: Send the Final Notification Either Way 29](#_Toc54007087)

[Step 13: Test the Whole Thing (Optional) 31](#_Toc54007088)

[A Final Word on Workflows 38](#_Toc54007089)

# Document History

|  |  |  |
| --- | --- | --- |
| Author | Version | Date |
| Matthew Trevathan | 1.0 | 20 Aug 2020 |
| Miguelángel Fernández | 2.0 | 03 Oct 2020 |

Comments:

|  |
| --- |
| {Add any comments here} |

# Module 2: Creating a Quantum Fabric Workflow

This lab module introduces the **Quantum** developer to the concept of **Fabric Workflows**.

In **Fabric**, a **Workflow** is an abstraction that aims to represent a business process using a subset of **BPMN**, which has been adapted to work well with other Fabric constructs.

Workflows are based on the **state transitions** of something. This something can be any entity within a business domain. For this particular lab, we'll be using an **application for a financial loan** as our business entity.

In Fabric, the business entity on which a Workflow is based is represented by a **Model** in an **Object Service**. Therefore, any Model used as the basis for a Workflow must have a field defined as its **workflow status**.

In summary, we'll define a model to represent an end user's application for a loan, and the changes in the state of this application will push it through the workflow.

## Goal

At the end of this lab you should be able to:

* Understand the basics of how workflows are designed and used.
* Build a workflow based on a pre-existing Object Service.
* Add pre-existing business rules to a workflow.
* Integrate a previously configured email service to a workflow.
* Test a published workflow.

## Tasks

At a very high level, this lab will guide you through the following tasks:

* Import a Fabric app which includes:
  + An Object Service called LoanApplicationObjS.
  + A Business Rule Set called LoanApprovalRules.
  + A Mock Integration Service called CreditBureauMock.
* Create a new Workflow based on the previously existing services mentioned above.
* Publish the newly created workflow.
* Test the newly created workflow.

## Required Software and Tools

### **Required:**

* A developer account at [manage.kony.com](http://manage.kony.com/)
* The file LoansOrigination1.zip containing the Fabric application we'll use for this lab.

### **Optional:**

**Only the steps in this lab titled with the word “Optional” will require the following:**

* **A Fabric Runtime Environment 9.1+:** In order to be able to publish a workflow you will need a paid-for Fabric account. If you do not have access to a paid-for account, you can still do this lab and follow along the steps to design a Workflow, but you will not be able to publish it.
* **Engagement Services 9.1+ with previously configured email** settings and an email address for testing.

Graphical user interface, application

Description automatically generated

# Module 2: Lab Instructions­

## Step 1: Import the Fabric App

1. Log into [manage.kony.com](http://manage.kony.com/).
2. Click on Apps option on the main menu to the left and click the IMPORT button.

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

1. Browse your file system and select the zipped Fabric bundle called LoansOrigination1(v1.0).zip as a new application.

Graphical user interface, text, application, PowerPoint

Description automatically generated

Graphical user interface, application

Description automatically generated

* **Note:** Make sure you give it a name that will tell it apart from the applications created by your peers during this lab —e.g.: Append your initials to the name of the application.

1. Verify that the new Fabric app LoansOrigination1 is imported.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Click the **Integration** tab and verify there’s a service called CreditBureauMock1(1.0).

Graphical user interface, text, application

Description automatically generated

1. Click the **Objects** tab and verify there’s a service called LoansApplicationObjS.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Click the **Rules** tab and verify there’s a service called LoanApprovalRules(1.0).

Graphical user interface, application

Description automatically generated

## Step 2: Create a Workflow

1. From your Fabric application go to the **Workflow** tab and click on the **CONFIGURE NEW** button.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Type in a name for your workflow —e.g.: LoansApplicationWF.

Graphical user interface, text, application, email

Description automatically generated

1. From the **Linked Object** dropdown select the **Use Existing** option.

Graphical user interface, application

Description automatically generated

1. From the list of existing object services, select the LoanApplicationObjS object service.

Graphical user interface, application

Description automatically generated

1. From the list of models in this object service, select the model called ApplicationModel.

Graphical user interface, application

Description automatically generated

1. From the list of fields in this model, select the status field and click the **ADD** button to save.

Graphical user interface, application, Teams

Description automatically generated

1. Verify that the model ApplicationModel appears as linked to the newly created workflow.

A picture containing chart

Description automatically generated

1. In the visual workflow editor, verify that a Start event and a User Task activity have been added to the workflow.

A picture containing text

Description automatically generated

## Step 3: Add a Task to Submit an Application

Let's create a way for an end-user (a potential borrower) to submit an application for a loan.

1. Click the User Task activity, click the **Properties** tab to the right and rename it to Submit Application.

Graphical user interface, application, Word

Description automatically generated

* **Note:** A **user task** is one that can later be surfaced —either through the API or a client application— for a user to interact with the workflow.

1. In the **Valid state transitions** field, set the value to New,Review,Rejected,Approved.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Add an End event to the workflow and connect the Submit Application task to it.

A picture containing graphical user interface

Description automatically generated

1. Click the **SAVE** button from the bottom right corner of the Console.

Graphical user interface, text, application, Word

Description automatically generated



**Pro Tip:** This is *not* really the end of the workflow. However, for a workflow to be in a *consistent* state, *every* possible flow must have an End. So provisionally adding an End event to every possible flow is just a good way to be able to save progress. We'll add more steps before the End node as we progress.

## Step 4: Get the Loan Applicant's Score

Let's query the credit score of the new potential borrower from a credit scoring bureau. For the sake of simplicity we will mock a credit scoring service using Fabric’s Mock Service Adapter.

1. Add a Service Task and place it just before the End event —you will have to disconnect the flow arrow from Submit Application task and add two new ones.

Diagram

Description automatically generated

1. Open the properties tab to the right and name the new task Get Credit Score.

Graphical user interface, diagram, application

Description automatically generated

1. Select Integration Service as the **Service Type**.
2. Select CreditBureauMock as the service linked.
3. Select getScore as the operation.

Graphical user interface, text, application

Description automatically generated

1. Click to configure the **Input Parameters**, select the credit bureau's id\_number input parameter and map it to the model's idDoc field from the BACKEND\_RESPONSE namespace.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

* **Note:** The BACKEND\_RESPONSE namespace represents the **Model** on which this workflow is based —i.e. in this case the ApplicationModel in the LoanApplicationObjS service.

1. Click to configure the **Output Parameters**, and type in credit as the namespace to store the output of this service. Click the **SAVE** button in the modal.

Graphical user interface, text, application, chat or text message

Description automatically generated Graphical user interface, text, application

Description automatically generated

* **Note:** This means that if —for example— this service responds with {"id": 1, "score": 725}, we can later refer to FABRIC\_WORKFLOW\_CONTEXT.redit.score and the result will be the value 725.

## Step 5: Update the Loan Application

Now let's update the loan application with the score we got from the credit bureau.

1. Again, add a Service Task and place it just before the End event. Reconnect the flow arrows accordingly.

Graphical user interface, application

Description automatically generated

1. Open the properties tab to the right and name it Update Application.

Graphical user interface, diagram

Description automatically generated

1. Select Object Service as the **Service Type**.
2. Select LoanApplicationObjS as the service linked.
3. Select PUT (Update) as the operation.
4. Click to configure the **Input Parameters**.

Graphical user interface, text, application

Description automatically generated

1. Map id from the BACKEND\_RESPONSE namespace to the id field in the model.
2. Map credit.score from the FABRIC\_WORKFLOW\_CONTEXT to the score field in the model.
3. Click the **SAVE** button in the modal and then click the **SAVE** button in the bottom right corner of the Console.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated

## Step 6: Test Our Progress So Far (Optional)

Let's just see how we're doing. We'll publish this to an actual environment and run it there.

1. From the Fabric application click the **"Publish"** tab at the top.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Click the checkbox next to an environment and click the **PUBLISH** button in the bottom right corner. Wait until the Console shows the status is Published. You will see the application's primary and secondary app keys and secrets listed below.

Graphical user interface, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

1. Click the **Workflow** icon just below the name of the environment. This will open a new tab to the Fabric Runtime's Console in the **Workflow Services** section and list the workflows published. You should see LoansApplicationWF in the list.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Click LoansApplicationWF. This will list all the instances of the ApplicationModel which have been pushed through the workflow. Each row shows the primary key of the model, its status, duration and relevant timestamps. Let's fire a new one.

Graphical user interface, text, application, email

Description automatically generated

1. Click the **Object Services** option from the main menu to the left side. This will show all the object services published. You should see the LoanApplicationObjS service.

Graphical user interface, application

Description automatically generated

1. Click the dropdown under the **App Data Model Objects** column in the LoanApplicationObjS row and select the ApplicationModel. This will display **Request Input** and **Request Output** tabs.

Graphical user interface, application

Description automatically generated

1. In the **Request Input** tab, select the **create** operation from the dropdown. This will invoke an HTTP POST request, so you will need a payload. Paste this in the **Body** of the request:

Graphical user interface, text, application

Description automatically generated

|  |
| --- |
| {  "idDoc": "123-1234-002",  "status":"New",  "firstName": "[your name]",  "lastName": "[your last name]",  "email": "[an email address for testing]"  } |

**Note:** The idDoc property is meant to be used to provide whatever alphanumeric national identification or fiscal identification number is relevant to Credit Scoring in each country —e.g.: The US’s SSN, Spain’s DNI and others like these.

1. Click the **Get Response** button. This will switch to the **Response Output** and once the request succeeds it should show the following response:

Graphical user interface, text, application

Description automatically generated

**Note:** The id property returned is auto-generated by the model. Take note of this value for the following steps.

1. Switch back to the **Request Input** and this time around select the get operation —which will fire an HTTP GETrequest— and again click the **Get Response** button. This time around the response should include a records property containing an array of all the existing loan applications.

Graphical user interface, text, application, email, Teams

Description automatically generated

1. Scroll down the body of the records array to the application for which the id matches the one in the response of the previous step. **Notice** that a new score property has been automatically added as a result of the workflow we've so far created.

Graphical user interface, text, application, email

Description automatically generated



**Pro Tip:** It's a good practice to build workflows by progressively adding and testing one or two tasks at the time. While this will require adding the End event in some provisional places and publishing each iteration, it is much more practical to use this approach than to try to build entire workflows all at once —especially large and complex ones.

## Step 7: Assess the Score and Update the Loan Application

Now that we have the score from the credit bureau, let's see if our business rules consider the applicant to be creditworthy or not.

We'll add a **Business Rule** that assesses whether the credit score is so good that the loan can be approved without further diligence, or whether it is so bad that it must be rejected right away. The rule will also determine that any score in between will require a credit analyst to have a closer look at the case.

1. Navigate to **Configure Services » Rules** and click on the LoanApprovalRules(1.0) rule set and then on the assessScore operation. The field **Rule Logic** shows how the rule is defined. Click on the **Expand** icon to examine it better. You may recognise the syntax is [**MVEL**](http://mvel.documentnode.com/), which in turn is based on **YAML**. Read more about Fabric Business Rules [here](https://docs.kony.com/konylibrary/integration/kmf_integrationservice_admin_console_userguide/Content/Rules_Services.htm).

|  |
| --- |
| ---  name: Poor  condition: score < 600  actions:  - results.addParam("decision", "Rejected")  ---  name: Good  condition: score >= 600 && score <= 699  actions:  - results.addParam("decision", "Review")  ---  name: Excellent  condition: score >= 700  actions:  - results.addParam("decision", "Approved") |

Graphical user interface, text, application

Description automatically generated

**Note:** The basic structure of **Business Rules** is that they can evaluate logical conditions on their input parameters, and execute actions based on those. In this case, the score input parameter is being evaluated in each rule’s condition against the ranges under 600, 600-699 and above 700. In each case, the actions for the rule add a decision to the output of the rule.

1. Go back to **Apps** in the Fabric Console, open the Fabric app you are working on, navigate to **Configure Service » Workflow** and click on the LoansApplicationWF workflow.
2. Add a Business Rule task between the Get Credit Score service task and the Update Application service task. Disconnect and reconnect the flow arrows accordingly.

Graphical user interface, application

Description automatically generated

1. Click the newly added business rule, open the properties tab to the right and name it Assess Score.
2. Select LoanApprovalRules(1.0) as the **Rule Set**.
3. Select assessScore as the business rule.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Click to configure the **Input Parameters**, add an input score from FABRIC\_WORKFLOW\_CONTEXT and type in credit.score. This takes the score we got from the credit bureau from the workflow's context and passes it to the business rule's score input parameter. Click **SAVE** to close the modal.

Graphical user interface, text, application

Description automatically generated

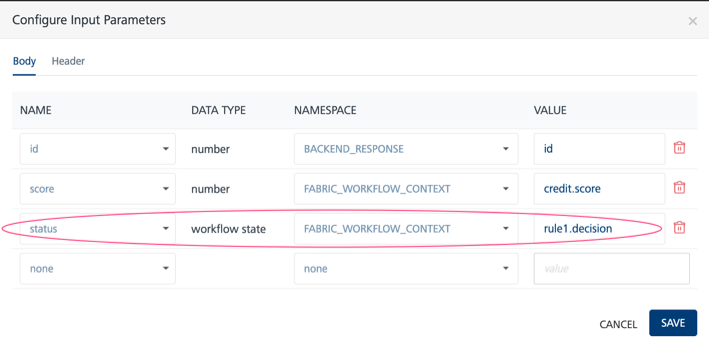
1. Click to configure the **Output Parameters**, and type in rule1 (or something else of your choosing) as the namespace to store the output of this service. Click **SAVE** to close the modal.

Graphical user interface, text, application

Description automatically generated

**Note:** This means that if this service responds with {"decision": "Approved", "opstatus": 0}, we can later refer to rule1.decision and get the Approved value.

1. Click the Update Application service task and click to configure the **Input Parameters**. You will see the mappings for the id and score fields which we added in a prior step.
2. Add a new mapping of rule1.decision from the FABRIC\_WORKFLOW\_CONTEXT to the status field in the model.



1. Click the **SAVE** button in the modal and then click the **SAVE** button in the bottom right corner of the Console.

## Step 8: Test Our Progress So Far (Optional)

If you wish, repeat the instructions in step #5 to publish and test your progress so far. If you do, you will see that every new loan application you create will have its status field updated to Approved, Review or Rejected, based on how the assessScore rule evaluates the score returned by the credit bureau.

## Step 9: Send an Approval Notification

If the applicant's score is good enough to approve right away, we should send out a notification.

1. Add an Exclusive Gateway just before the End event and reconnect the arrow flows as required.

Graphical user interface, diagram

Description automatically generated

1. Click the new gateway, click the **Properties** tab to the right and name it Gateway1.

Graphical user interface, diagram, application

Description automatically generated

1. Add a Message event onto the canvas right between Gateway1 and the End event. Click the properties tab to the right and name it Notify Approval.

Graphical user interface, diagram, application

Description automatically generated

1. Select the new Notify Approval message event, click the arrow, and from the **Properties** tab to the right, name it isApproved and click the **ADD** button under **Entry Validation Criteria**.

Diagram

Description automatically generated

1. Click the **Add Condition** button, select FABRIC\_WORKFLOW\_CONTEXT, type rule1.decision for the left-hand value, select the == operator, select none for the context and type Approved for the right-hand value, and click the **SAVE** button on the modal.

Graphical user interface, text, application

Description automatically generated

1. Click on the Notify Approval event, click the **COMPOSE EMAIL** button from the **Properties** tab to the right and set the following values.

|  |  |
| --- | --- |
| **To** | $email |
| **Subject** | Your loan has been approved! |
| **Body** | Dear $firstName,<br>We're pleased to inform you that your loan has been <b>approved!</b> |

Graphical user interface, text, application, email

Description automatically generated

1. Click the **Parameters** tab on the right side of the modal and add the following mappings and click the **SAVE** button on the modal:

* Map email from BACKEND\_RESPONSE to email.
* Map firstName from BACKEND\_RESPONSE to firstName.

Graphical user interface, text, application

Description automatically generated

## Step 10: Send a Rejection Notification

If the applicant's score is bad enough to approve right away, we should send out a notification.

1. Add another Message event onto the canvas. Click the properties tab to the right and name it Notify Rejection.

Graphical user interface, diagram

Description automatically generated

1. Drag a new arrow from Gateway1 to the new Notify Rejection message event, click the arrow, and from the **Properties** tab to the right, name it isRejected and click the **ADD** button under **Entry Validation Criteria**.

Diagram

Description automatically generated

1. Click the **Add Condition** button, select FABRIC\_WORKFLOW\_CONTEXT, type rule1.decision for the left-hand value, select the == operator, select none for the context and type Rejected for the right-hand value. Click the **SAVE** button on the modal.

Graphical user interface, text, application

Description automatically generated

1. Click on the Notify Rejection event, click the **COMPOSE EMAIL** button from the **Properties** tab to the right and set the following values.

|  |  |
| --- | --- |
| **To** | $email |
| **Subject** | Your loan has been rejected |
| **Body** | Dear $firstName,<br>We regret to inform you we <i>cannot</i> approve your loan application at this time. |

1. Click the **Parameters** tab on the right side of the modal and add the following mappings and click the **SAVE** button on the modal:

* Map email from BACKEND\_RESPONSE to email.
* Map firstName from BACKEND\_RESPONSE to firstName.

Graphical user interface, application

Description automatically generated

1. Add an End event just after the Notify Rejection event and click the **SAVE** button on the bottom right corner.

Diagram

Description automatically generated

## Step 11: Make Room for a Human to Decide

If the applicant's score is neither too good nor too bad to make an automatic decision, we need human intervention —e.g.: from a credit analyst— to decide.

1. Add a new User Task activity, click the **Properties** tab and name it Review Case.

Diagram

Description automatically generated

1. To specify that after reviewed an application must be either approved or rejected, set the **Valid state transitions** field to Approved, Rejected.

Graphical user interface, diagram, application

Description automatically generated

1. Connect Gateway1 to the newly added Review activity. Click the connecting flow, click the **Properties** tab to the right, name it needsReview and click the **ADD** button under **Entry Validation Criteria**.

Graphical user interface, diagram, application

Description automatically generated

1. Click the **Add Condition** button, select FABRIC\_WORKFLOW\_CONTEXT, type rule1.decision for the left-hand value, select the == operator, select none for the context and type Review for the right-hand value. Click the **SAVE** button on the modal.

Graphical user interface, text, application

Description automatically generated

* **Note:** With this, we've covered all the three possible scenarios for Gateway1 based on the value of the application model's status field.

## Step 12: Send the Final Notification Either Way

Let's reuse the existing Message events.

1. Add a second Exclusive Gateway, name it Gateway2 and connect the Review Case task to it.

Graphical user interface, application

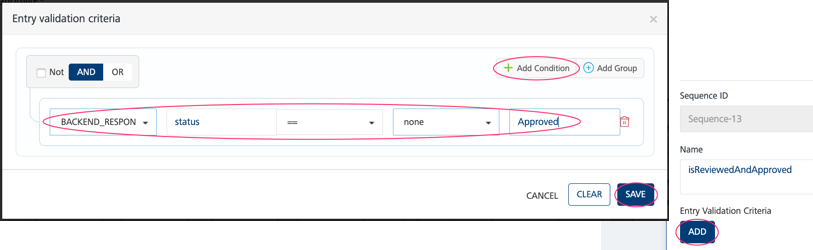
Description automatically generated

1. Connect Gateway2 to the already existing Message event called Notify Approval. Select the new arrow, click the **Properties** tab to the right and name it isReviewedAndApproved.

Diagram

Description automatically generated

1. Click the **ADD** button under **Entry Validation Criteria**, click the **Add Condition** button, select BACKEND\_RESPONSE, type status for the left-hand value, select the == operator, select none for the context and type Approved for the right-hand value. Click the **SAVE** button on the modal.



1. Connect Gateway2 to the already existing Message event called Notify Rejection. Select the new arrow, click the **Properties** tab to the right and name it isReviewedAndRejected.

Graphical user interface, diagram, application

Description automatically generated

1. Click the **ADD** button under **Entry Validation Criteria**, click the **Add Condition** button, select BACKEND\_RESPONSE, type status for the left-hand value, select the == operator, select none for the context and type Rejected for the right-hand value. Click the **SAVE** button on the modal.

Application

Description automatically generated

1. Verify the end-to-end workflow looks like the one below and click the **SAVE** button on the lower-right corner.

Diagram

Description automatically generated

## Step 13: Test the Whole Thing (Optional)

Let's first test a loan application by a borrower with an excellent credit score. In this case the application will be approved automatically. Then, we'll test another loan application by a borrower with a poorer score. A credit analyst will then manually reject the application.

1. Click the **Publish** tab under the LoanOrigination1 Fabric app, check the box next to an environment and click the **PUBLISH** button in the bottom right corner. Wait until the Console shows the status is Published. You will see the application's primary and secondary app keys and secrets listed below.

Graphical user interface, website

Description automatically generated

1. Click the **Object Services** icon just below the name of the environment. This will open a new tab to the Fabric Runtime's Console in the **Object Services**. This will show all the object services published. You should see the LoanApplicationObjS service.

Graphical user interface, application

Description automatically generated

1. Click the dropdown under the **App Data Model Objects** column in the LoanApplicationObjS row and select the ApplicationModel. This will display **Request Input** and **Request Output** tabs.

Graphical user interface, application

Description automatically generated

1. In the **Request Input** tab, select the create operation from the dropdown and paste this in the **Body** of the request:

|  |
| --- |
| {  "idDoc": "123-1234-002",  "status": "New",  "firstName": "Tywin",  "lastName": "Lannister",  "email": "[an email address for testing]"  } |

Graphical user interface, text, application

Description automatically generated

1. This will return a response that looks like the one below. Make a note of the id value.

|  |
| --- |
| {  "id":"[number]","opstatus":0,"httpStatusCode":0  } |

Graphical user interface, text, application

Description automatically generated

1. Repeat step #5 and #6 with this payload:

|  |
| --- |
| {  "idDoc": "123-1234-005",  "status": "New",  "firstName": "Jon",  "lastName": "Snow",  "email":"[an email address for testing]"  } |

1. Now select the get operation and click the **Get Response** button to verify that both applications have been created. You’ll notice that Tywin has such a good score that his application has been approved automatically. Whereas Jon does not have such a good score and so the case needs further review.

Graphical user interface, text, application, email

Description automatically generated

1. Click the **Workflow Services** option from the main menu to the left, and then select LoansApplicationWF. You should see two newly fired workflow executions for each of the loan application you've just created. The first one —Tywin's— will be COMPLETED. The second one —Jon's— will be RUNNING. The RUNNING status of Jon’s application means it’s still waiting for a human to take action.

Graphical user interface, application, Teams

Description automatically generated

1. Out of the two newly fired workflow executions, click on Tywin's application. You should see that the status for all the actions is DONE. If the execution is not done, it might be still running but will finish shortly. Click the **Refresh** button.
2. Graphical user interface, table

   Description automatically generated

* **Note:** If the notification step shows FAILED, go back and check your workflow is well defined, and make check the **Email Configuration** in your **Engagement Services**.

1. Go back to the list of the executions of LoansApplicationWF and click on Jon's application. You should see that the status of the last action is PAUSED. This means that the workflow is waiting for a human to intervene.

Graphical user interface, table

Description automatically generated

1. Now let's simulate a credit analyst's approval. Click the **Object Services** option from the main menu to the left, click the dropdown under the **App Data Model Objects** column in the LoanApplicationObjS row and select the ApplicationModel.
2. In the **Request Input** tab, select the **update** operation from the dropdown —This will fire an HTTP PUT operation. Paste this in the **Body** of the request:

|  |
| --- |
| {  "id": [the id of Jon's application],  "status": "Rejected",  "firstName": "Jon",  "lastName": "Snow",  "email":"[an email address for testing]"  } |

Graphical user interface, text, application

Description automatically generated

1. This will return a response saying that one record has been updated.

Graphical user interface, text, application, email

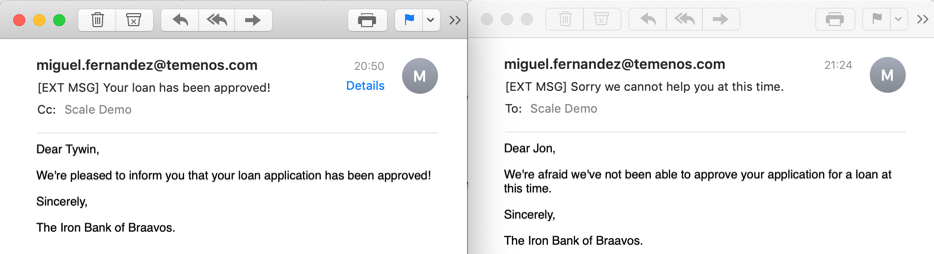
Description automatically generated

1. Go back to **Workflow Services** and click LoansApplicationWF. You should now see Jon's application is also in COMPLETED status. Click on it and verify that all activities are in DONE status.

Graphical user interface, application, Teams

Description automatically generated

1. Check the inbox of the email address you're using for testing. You should have received emails notifying you of the results of both loan applications.



# A Final Word on Workflows

During this lab you’ve designed, published and tested a workflow. As you will have been able to appreciate, workflows are advanced tools that build on top of other Fabric constructs, such as Integration Services, Object Services, Business Rules and Engagement Services. If you’ve struggled with these concepts we recommend you go back and explore the labs and other enablement materials that go over these other features in more detail.

Also, workflows are very powerful tools that will allow you to decouple client apps from business processes. If used right, Fabric Workflows could allow you to change processes without affecting the client applications or having to distribute new versions of them. They could also allow you to start a user journey on one device, continue on a second device and finish on a third one.

However, workflows should not be abused. Not everything in your app is necessarily a good candidate to be modelled as a Fabric Workflow. Some use cases may be too simple or trivial to become workflows. Meanwhile, others may be way too large or complex. Fabric Workflows are meant as a lightweight solution for use cases which commonly arise when building multi-experience digital experiences, but not as a fully-fledged BPMS for the entire enterprise.