Connecting a Quantum App to a Temenos Microservices via Object Services

**Lab Guide**



# Table of Contents

[1](#_Toc51140737)

[Table of Contents 3](#_Toc51140738)

[Document History 4](#_Toc51140739)

[Lab Environment Overview 5](#_Toc51140740)

[Installed Software and Tools 5](#_Toc51140741)

[Module 2: Connecting an Object Service 6](#_Toc51140742)

[Module 1: Lab Summary 22](#_Toc51140743)

# Document History

|  |  |  |
| --- | --- | --- |
| Author | Version | Date |
| Matthew Trevathan | 1.0 | 8.20.2020 |

Comments:

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

# Lab Environment Overview

# Installed Software and Tools

|  |  |
| --- | --- |
| Software | Link |
| Visualizer 9 SP1 | https://community.kony.com/downloads |
| Kony Fabric | Manage.kony.com |
|  |  |
|  |  |

# 

# Module 2: Connecting an Object Service

|  |  |
| --- | --- |
| Purpose: | This lab introduces object services.  After completing the lab, you should be able to:   * Create a Fabric app * Build an integration service with Fabric * Create an Object Service   Connect the client to the object service   * Deploy a fabric app * Test the client. |
|  |  |
| Tasks: | Tasks you will complete in this lab exercise include:   * + Import a Visualizer App   + Create a new Fabric app   + Create an integration service   + Add an operation   + Create an Object service   + Testing your object service   + Deploy the Fabric app   + Connect your object service to your client app   + Run Live Preview |

**Module 2: Lab Instructions**

| Step | Action |
| --- | --- |
| 2 | **Create a New Fabric App**   1. Open Visualizer 2. Click the Fabric icon in the left side navigation.   A close up of a sign  Description automatically generated   1. Your client app should already be linked to a Fabric app since a fabric app is automatically created when we created our identity service in the last module. Here is my BankingDemoApp. Note- IF you create a new app, Fabric defaults the name of the Fabric app to match your Visualizer app.   A screenshot of a cell phone  Description automatically generated |
| 3 | **Create New Integration Service**   1. If your integration tab isn’t selected, select the Integration tab.   A screenshot of a cell phone  Description automatically generated   1. Click Configure New.   A screenshot of a cell phone  Description automatically generated   1. Enter MarketServices for the Name. This will be the name of the integration service, and it represents the request and response to the back end that we will map to the object model later in the module. 2. Quantum Fabric allows users to choose from a variety of back end services to expose to the end user, including JSON,XML,SOAP and many others. This allows a developer to select many different types of integrations and expose them as RESTful JSON services to end user experiences. Select JSON for the service type in the dropdown. 3. Enter  <https://ase3skcx80.execute-api.us-east-1.amazonaws.com/stage-scale/> for the base URL. This points to the Temenos microservice base marketing microservices. 4. Click Save and Add Operation. |
| 4 | **Add an Operation**   1. In the Name text field, type Products.   A screenshot of a cell phone  Description automatically generated   1. In the suffix field, add the following: /v1.0.0/product/products 2. For this operation, we need to fetch some data, so we will keep the target HTTP method as GET. 3. In the Request Input section, select Header. It’s very important your select the Header   Graphical user interface, text, application, chat or text message  Description automatically generated   1. Select the Add Parameter button. We need to add an API key in the session header. 2. For the name of the key, type x-api-key. 3. For the default value, type VkWTvp5H6F6FwiGIYFDzh56dDv4I1bv0akVrrjhk . This value is the key for the Temenos Marketing Microservice.   A close up of a logo  Description automatically generated   1. Click the Save and Fetch Response button at the bottom of the screen. This will return a backend response.   A screenshot of a social media post  Description automatically generated   1. Check to make sure your Backend Response returned a JSON object in the Backend Response pane that is visible after you clicked the Save and Fetch Response button. If there isn’t a response. Click the Save and Fetch Response button a second time. Notice at this point, there isn’t any output in the output result. Filter the backend response into our output result that the integration service response with.   A screenshot of a cell phone  Description automatically generated   1. From the center panel, find the products node and right click the node and click Create response.   A screenshot of a cell phone  Description automatically generated   1. This will fill the output response inside the service mapping with all the attributes within the product node, including any arrays of objects on any sub-nodes. We don’t need all the information provided, so let’s remove some of the data to create a concise response. Let’s remove all the data below the extensionData by clicking each field from Extension data to the last field in the response and click delete.   A screenshot of a cell phone  Description automatically generated   1. Once the attributes are removed, the screenshot above shows the remaining attributes we want to keep. Click Save and Fetch.   A screenshot of a cell phone  Description automatically generated   1. Look in the output result panel. Now, the result returns the attributes we filtered from the backend service call to create a concise response for our multi-experience apps. Now, we have an integration service that we can connect to our app. In the next section, we will decouple the backend integration service with our client app using an object service. |
| 5 | **Create an Object Service**  **A screenshot of a cell phone  Description automatically generated**   1. Select the Objects tab.   A close up of a sign  Description automatically generated   1. Select Configure New.   A screenshot of a cell phone  Description automatically generated   1. Name your Object service Marketing. 2. Select Integration & Orchestration Service for the endpoint type. There are a number of other services we can map directly into an object that you can see if you pull down the drop down, included mapping to relational databases. In our case, we created an integration service to the backend we want to map to, so we select the Integration & Orchestration Service. 3. Click the Save and Configure button.   A screenshot of a cell phone  Description automatically generated   1. At this point, we have an object service without any object. Notice in the left side panel, we don’t have a data model yet. The data model defines how a client app maps request and response data into the Fabric app. You can configure your own data model, even if the back end doesn’t exist by clicking the configure New button, but let’s generate a data mode from our existing Integration service. Click the Generate button.   A screenshot of a cell phone  Description automatically generated   1. Next, create an object named MKTProducts. 2. Let’s use the service we created in the last step/ Select your MarketingServices and find the Product operation. 3. Click Generate.   A screenshot of a cell phone  Description automatically generated   1. Now, our data panel has a data model. Click the mapping tab.   A screenshot of a cell phone  Description automatically generated   1. Now, we need to map our backend service to our data model. Click the productDescriptiopn in the left 2. Let’s map the GET verb to our service. Select GET from the Data Model Verb drop down. 3. Select the MarketServices service from the Services dropdown. 4. Select product from the Operations dropdown 5. Click the add mapping button.   A screenshot of a cell phone  Description automatically generated   1. The Fabric console displays the current mapping between our prodDescription object and our backend services. Click the Response Mapping tab. 2. Click the Edit button.   A screenshot of a social media post  Description automatically generated   1. Drag the mapping from the Service on the left to the object on the right until each attribute is mapped. 2. Once all the attributes are mapped, click save. |
| 6 | **Testing our Object Service**   1. Now, we can test the service. Click the Test Tab.   **A screenshot of a cell phone  Description automatically generated**   1. Click Send.   A screenshot of a social media post  Description automatically generated   1. When we send the request to the object service, the object service calls the integration service, and the response is mapped back into the response of the object service. 2. Now that the services are configured, let’s deploy our app. |
| 7 | **Deploy Fabric App**   1. Select the Publish tab.   A screenshot of a cell phone  Description automatically generated   1. The Publish tab shows all the available environments you can publish to within your account. In my case, I only have one environment to publish to, XAI POC. Click the circle beside the instance you want to deploy to. When you do, A checkmark will appear for that instance. 2. Click the Publish button. 3. When the application is published, the publication window will show a green checkmark.   A screenshot of a cell phone  Description automatically generated   1. Close your browser, and open your Visualizer app. |
| 8 | **Connecting the Object Service to Your App**   1. Let’s pick up our app where we left off on the identity module. Select the Data & Services Tab. 2. Hit the Refresh button   A screenshot of a cell phone  Description automatically generated   1. Notice from this screen, we can see both the integration service and the object service. If we deleted the integration service at this point, the object service would remain, and we could map a new service to the object service without needed to remap the service to the client app. If we chose the integration service here, and changed the service, we would need to map the client to the new service that may now have the same field names that the previous service did.   A screenshot of a cell phone  Description automatically generated   1. In the left side navigation, click Responsive Web 🡪 Forms 🡪 frmTerms. This will bring up the terms and conditions page.   Graphical user interface, text, application  Description automatically generated   1. In the right hand panel, make sure you’re on Panel, make sure you’re on Data and Services. 2. Find your object service you created in the previous steps. If there aren’t any services when you expand the Project Services tab, hit the refresh button and it will refresh the services. 3. Expand the project services and click the ProdDescription microservice we created.   Text  Description automatically generated   1. Drag the get operation onto your terms and conditions form.   Graphical user interface  Description automatically generated   1. Once you drag the get operation on the screen, Visualizer will give you 3 options for mapping the data. The first option will autogenerate a List & Details template to show the data, the second will present the data in an autogenerated grid layout. The third options will let us map the data from the response to the form. Choose Detail Using Response. 2. Now, drag the detailedDescription from the data panel into the first text box under the Advance Checking Label.   Graphical user interface, text, application  Description automatically generated   1. Now drag the termsConditions attribute onto the second text box. 2. If you want to see the client side workflow generated by Visualizer, select the properties in the right hand tab.   Calendar  Description automatically generated   1. Select Edit on the onMapping action. This will bring up the workflow.   A screenshot of a computer screen  Description automatically generated   1. You can see that Visualizer created a number of workflow steps. If you want to edit any of the steps, you can easily edit the workflows from here. 2. Hit cancel or save. Now, let’s look at our app in Action. |
| 9 | **Run the App in Live Preview**   1. From the top menu, select Build-> Live Preview Settings   A screenshot of text  Description automatically generated   1. Ensure the responsive web checkbox is selected   A screenshot of a cell phone  Description automatically generated   1. Click Save and Run. 2. This will locally compile your app as a responsive web app and start the preview browser.   Graphical user interface, application  Description automatically generated   1. Login to the application suing the credentials you created in the first part of this lab in your user repository. 2. If you are successful, you will successfully navigate to the dashboard form. 3. From the dashboard, click the advertisement, and it will load the terms page with the text fields filled with data from our object service. |

## Module 2: Lab Summary

In this module, we created an integration service that exposes a backend service as a REST based service for application. We also learned how to decouple our backend integration service from our front in client app by building and deploying an object services that we mapped to our client app. In the next module, we will learn how to create a light weight workflow with Fabric workflow engine.