

iPaaS Workshop

Lab Instructions



Table of Contents

Introduction	4
Overview	4
1. Validate Your Environment.....	5
1.1 Sign up for Cloud Trial Environment.....	5
1.2 Sign in to TIBCO Cloud.....	7
1.2.1 Getting Ready.....	7
1.2.2 How to Do It	7
2. Design an API and Create a Mock Application.....	10
2.1 Design an API	10
2.1.1 Getting Ready.....	10
2.1.2 How to Do It	11
2.2 Create a Mock App.....	15
2.2.1 Getting Ready.....	15
2.2.2 How to Do It	16
2.3 Import API Specs	17
2.3.1 Getting Ready.....	17
2.3.2 How to Do It	18
2.4 Additional Reading.....	22
3. Build and Deploy Your API	23
3.1 BusinessWorks Apps in TIBCO Cloud Integration	23
3.1.1 How to Do it: Import the Business Works Application via EAR import.....	23
3.2 Flogo Apps in TIBCO Cloud Integration.....	27
3.2.1 Getting Ready.....	27
3.2.2 How to Do It: Create the Skeleton Flogo App from the API Specification	27
3.2.3 How to Do It: Implement the Flogo App.....	29
3.2.4 Push the Integration App to TIBCO Cloud and Test It.....	38
3.3 Additional Reading	42
Extra LAB: Business Events Simple Decision Service.....	43
What You Will Learn	43
Getting Ready.....	43
Creating a Base TCE Project	44
Implementing a Simple decision service.....	46

Deploying the Decision Service	49
Testing the Decision Service	51
Summary	52
Extra Lab: BusinessWorks Apps in TIBCO Cloud Integration	53
1 Getting Ready.....	53
2 How to Do It: Connect Business Studio to TIBCO Cloud	53
3 How to Do It: Import a Project in to Business Studio	55
4 How to Do It: Push the Project to TIBCO Cloud and Test	56

Introduction

In this lab exercise you will learn the Tibco Cloud Integration capabilities. You will define new API's in the API management by an API-Led approach. Then you will implement this API using the Integration capabilities. For the connection to the Business Applications, you will deploy existing on-premise Business Works integrations into the Tibco Cloud.

Overview

The use case that you will build is a sales order processing scenario.



In a step by step lab instruction, you will learn the Tibco Cloud Integration service and experience the low code development capabilities of our platform.

1. Validate Your Environment

In this section, you'll validate your lab environment, which consists of a TIBCO Cloud Integration trial environment.

1.1 Sign up for Cloud Trial Environment

You need access to TIBCO Cloud Events subscription. If you do not have one, you can request for free trial via the below link.

TIBCO Cloud™: <https://account.cloud.tibco.com/signup/tci>

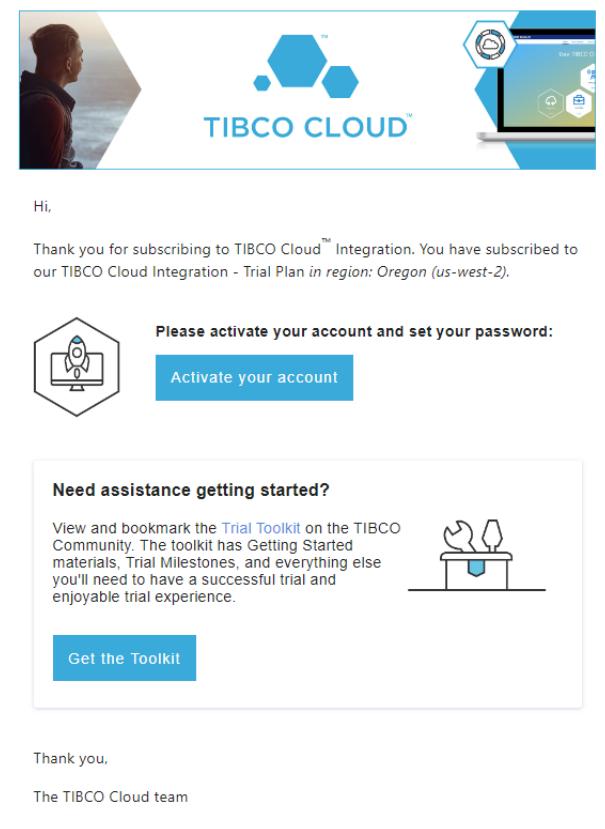
The screenshot shows the TIBCO Cloud Integration trial sign-up interface. At the top, there's a banner with a cityscape background and the text "Sign up for a 30-Day Free Trial of TIBCO Cloud™ Integration". Below the banner, it says "Choose your hosting environment below" with options for "AWS" (selected) and "Azure". The main form fields include:

- Email: michelle.leijdekker@ziggo.nl
- First Name: Michel
- Last Name: Leijdekker
- Phone Number: +31 627229441
- Product Use: Business use (selected)
- Company: Tibco
- Role: Solution Engineer
- Location: Netherlands
- Region: AWS United States

At the bottom, there are checkboxes for agreeing to the End User License Agreement and Privacy Policy, and for opting-in to receive contact from TIBCO. A large blue "START FREE TRIAL" button is at the bottom right.

- ❖ choose the “AWS United States” as a region and click “Start Free Trial”

- ❖ Wait for the Welcome email to activate your trial account.



- ❖ Click the "Activate your Account" button and set your password in the form.

Create your password

.....

Password policy ▾

- ✓ Must have minimum 8 characters.
- ✓ Include atleast three of following
 - English uppercase characters (A through Z)
 - English lowercase characters (a through z)
 - Non-alphanumeric characters (such as !, @, #, \$, %, ^, &, * etc)
 - Numbers (0 through 9)
- ✓ Password mismatch

Confirm password

.....

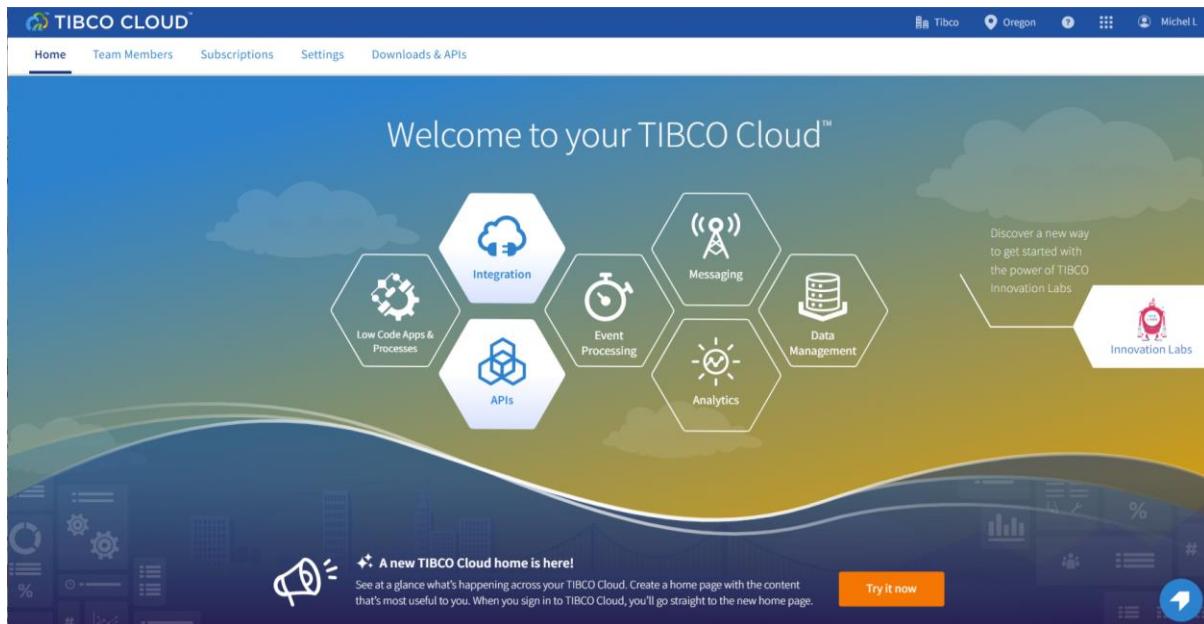
I have read and accept the [End User License Agreement](#), [Privacy Policy](#) and [Terms of Services](#).

ACTIVATE

1.2 Sign in to TIBCO Cloud

1.2.1 Getting Ready

This lab assumes you have started a trial of TIBCO Cloud™ Integration (If you haven't, [click here](#) and follow the instructions below)



1.2.2 How to Do It

1. Browse to <https://cloud.tibco.com/>.

A screenshot of the TIBCO Cloud landing page. At the top, there's a dark header with the TIBCO Cloud logo and a navigation menu with links for Overview, Platform Capabilities, Cloud Composer, Status, SIGN IN, and TRY NOW. Below the header, a main section has a dark background with a network graph overlay. It features the text 'The power of TIBCO, in the cloud' and a bulleted list: '• Unified Experience', '• Security by Design', and '• Enterprise Grade'. There are 'SIGN IN' and 'LEARN MORE' buttons. Below this, a section titled 'TIBCO Cloud: Connected Intelligence in Action' has a sub-section titled 'TIBCO Cloud™ is the digital platform that runs and adapts your connected business'. At the bottom, there's a graphic showing three overlapping circles labeled 'CONNECT' (blue), 'UNIFY' (green), and 'PREDICT' (orange).

2. Click **SIGN IN** (at the right top) and select the **United States – West** Region

Select the region for your TIBCO Cloud™ Account subscription

United States - West (Oregon)	powered by 	Continue
United States - East (North Virginia)	powered by 	Continue
Europe (Ireland)	powered by 	Continue
Australia (Sydney)	powered by 	Continue
United States (Washington)		Continue

3. Fill out the relevant account details (**Email Address** and **Password**) in the following screen.



Sign into TIBCO® Account
To continue to [TIBCO Cloud](#)

Email Address

Password

[TIBCO LOGIN](#)

Need help with Login?

OR

 [Use your corporate account](#)

 [Sign in with Google](#)

- After this, you should see a landing page which looks similar to this:



2. Design an API and Create a Mock Application

API Modeler is a web-based tool that provides you with the capability to graphically create and model a REST API.

With API Modeler, you can import REST API specifications either in a YAML or JSON file for further editing, or model your own REST API step-by-step in a visual interface. After modelling an API, you can choose to mock it and see the API in action, or directly implement it.

In this section, you'll design an API using API Modeler and generate a mock application based on it. Furthermore, you'll import several API specs.

- Design an API
- Create a Mock App
- Import API Specs

2.1 Design an API

2.1.1 Getting Ready

In this lab, you'll create an API using the API Modeler. To navigate from the landing page to the API Modeler, do the following:

1. Navigate from the TIBCO Cloud landing page to Cloud Integration by clicking the **APIs** hexagon.
2. Click on the **API Model and Mock** link.
3. After this, your screen should look similar to this:

The screenshot shows the TIBCO CLOUD Integration interface. On the left, there's a sidebar titled "All API specs (1)" under "GROUPS (1)". A modal window is open over the main content area, prompting the user to enter a "New group name" (with "MyTCIWorkshop" typed in) and a "Description". The main content area shows a table of API specifications, with one entry visible: "TIBCO Cloud Integration PetStore... 1.1". The table includes columns for "API name", "Version", "Description", "Group", and "Last modified".

2.1.2 How to Do It

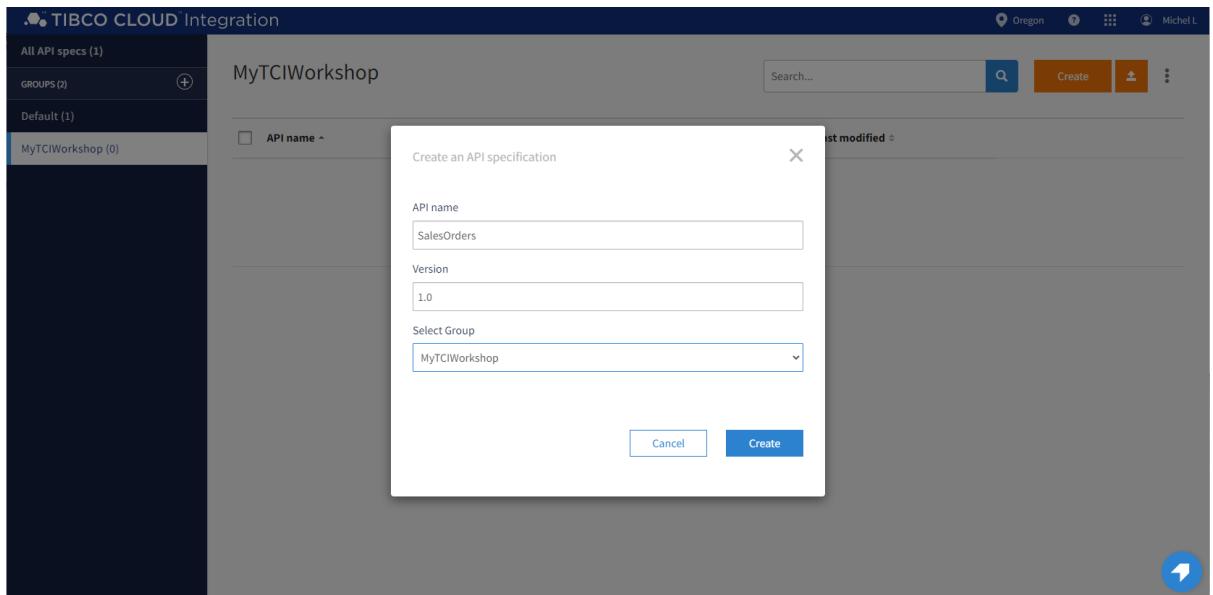
1. Create a group (this is optional) by clicking the + sign to the right from **GROUP** and give it a name e.g. **MyTCIWorkshop**.

This screenshot shows the same interface as above, but the modal window for creating a group has been closed. Instead, a new modal window is open, showing the creation of an API spec. The "Select Group" dropdown is set to "MyTCIWorkshop". The form fields include "API name" (SalesOrders), "Version" (1.0), and "Select Group" (MyTCIWorkshop).

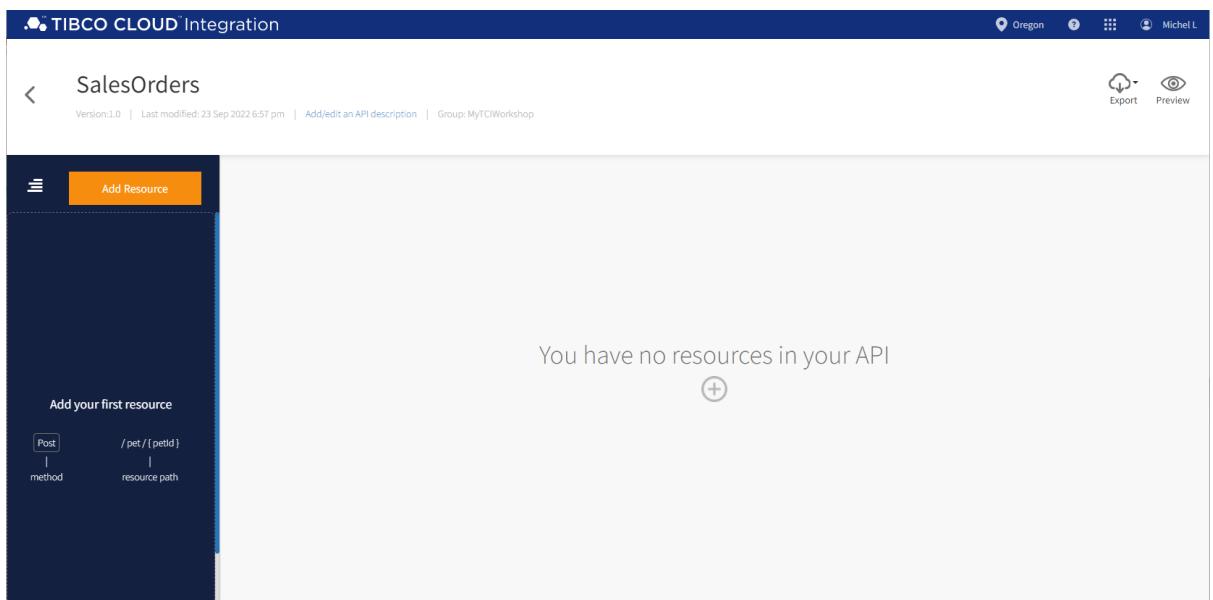
2. Create an API spec by clicking on **MyTCIWorkshop**, and click on the orange **Create** button.
3. In the form, use the following values:

Field	Value
API name	SalesOrders
Version	1.0
Select Group	MyTCIWorkshop

4. It should look something like this:



5. Click the blue **Create** button
6. Add a resource to the API spec by clicking on the orange **Add Resource** button in the next **SalesOrders** screen.



7. In the form, use the following values:

Field	Value
Resource path	/sync/{orderId}
Method	POST

8. The form should look something like this:

The dialog box has a title 'Add a resource and methods' and a close button 'X'. It contains a text input field 'Enter a new resource path (begin with /)' with the value '/sync/{orderId}'. Below it is a section for adding methods with buttons for 'GET', 'POST', 'PUT', 'PATCH', and 'DELETE'. The 'POST' button is highlighted in green. At the bottom are 'Cancel' and 'Save' buttons.

9. Click on the blue **Save** button.

10. Edit the success response of the **POST** method by first clicking on the white **Response** button in the next **POST /sync/{orderId}** screen, and then clicking on the **Success response (200/OK)** card.

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'TIBCO CLOUD' integration, 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. The main area displays the 'SalesOrders' API version 1.0. The 'POST /sync/{orderId}' endpoint is selected. On the right, there are 'Request' and 'Response' tabs. Under 'Response', the '200/OK' card is selected, and there is an 'Add' button to create a new response card.

11. In the form, click on the **Generate Schema from Sample Data** link.

The screenshot shows the 'Edit Response' dialog for the 200 OK response. It includes fields for 'http status code' (set to 200) and 'Description' (set to 'Success response'). Below these are sections for 'Headers' and 'Schema'. A red box highlights the 'Generate Schema from Sample Data' link at the bottom right of the schema section.

Then on the orange **Continue** button.

Copy the below sample json in the **Provide Sample Data** field:

```
{  
  "_response_string": "Order has been created with SalesDocumentID 00000013572"  
}
```

The form should look something like this:

The screenshot shows a configuration interface for a service operation. At the top, there's an 'Edit Response' section with fields for 'http status code' (set to 200) and 'Description' (set to 'Success response'). Below this is a 'Headers' section with a note: 'Your operation has no header objects'. The main area is titled 'Provide Sample Data' and contains a code editor with the following JSON content:

```
1 {  
2   "_response_string": "Order has been created with SalesDocumentID 00000013572"  
3 }
```

Next to the code editor are two buttons: 'Generate Schema' (orange) and 'Input Schema' (green). At the bottom right of the interface are 'Cancel' and 'Save' buttons.

Click on the orange **Generate Schema** button, and replace **GiveNewSchemaNameHere** by **salesOrderSyncResponse** in the next form.
The form should look something like this:

The screenshot shows the 'Edit Response' dialog in the TIBCO Cloud Integration interface. It includes fields for 'http status code' (set to 200) and 'Description' (Success response). The 'Headers' section is empty. The 'Schema' section shows 'salesOrderSyncResponse' as an array type. A JSON preview pane displays the following sample response:

```

1: {
2:   "type": "object",
3:   "properties": {
4:     "_response_string": {
5:       "type": "string",
6:       "default": "Order has been created with SalesDocumentID 00000013572"
7:     }
8:   }
9: }

```

At the bottom right are 'Cancel' and 'Save' buttons.

Click on the blue **Save** button.

12. Click on the **Preview** icon, and your screen should look similar to this:

The screenshot shows the 'API Specs' page for the SalesOrders resource. It displays the 'Preview' view for the POST /sync/{orderId} operation. The response class is listed as 'Success response' with the model 'Example Value' shown as:

```

{
  "application/json": {
    "_response_string": "Order has been created with SalesDocumentID 00000013572"
  }
}

```

The 'Parameters' table shows one parameter:

Parameter	Value	Description	Parameter Type	Data Type
orderId			path	string

At the bottom left is a note: '[BASE URL: , API VERSION: 1.0]'

2.2 Create a Mock App

2.2.1 Getting Ready

In this lab you'll create a mock application based on the API specification created in the previous lab. In order to do this:

1. Navigate to the API specifications by clicking on the **API Specs** menu item.

- Select the group you've created the **SalesOrders** API specification in, e.g. **MyTCIWorkshop**.
- Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'TIBCO CLOUD' logo, location 'Oregon', and various icons. The main menu has tabs for 'Apps', 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. On the left, a sidebar titled 'All API specs' shows a 'GROUP' section with 'Default' and 'MyTCIWorkshop' selected. The main content area is titled 'MyTCIWorkshop' and lists the 'SalesOrders' API specification. The table columns are 'API name', 'Version', 'Description', and 'Last modified'. A search bar and a 'Create' button are at the top right of the list.

2.2.2 How to Do It

- Hover over the **SalesOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:

This screenshot is similar to the previous one, showing the 'MyTCIWorkshop' group. However, a context menu is open over the 'SalesOrders' row. The menu items include 'Create Mock app' (with a circled red arrow pointing to it), 'Generate Node.js code', 'Create Flugo app', 'Clone', 'Edit', and 'Remove'.

- Create a mock app by clicking on the **Create** button in next form:

The dialog box is titled 'Create Mock app' and contains a single input field labeled 'Give your new Mock app a name' with the value 'salesorders_1_0_mock_app'. At the bottom are 'Cancel' and 'Create' buttons.

- Once the mock app is running, hover over the **Endpoint** link, and select **View and Test** from the menu:

The screenshot shows the TIBCO CLOUD Integration interface. At the top, there's a navigation bar with the TIBCO logo, location (Oregon), user (Michel L.), and a three-dot menu. Below the bar, the application title is 'Salesorders_1_0_mock_app', with status 'Modified on 23 Sep 2022 7:34 pm' and 'Owner: Michel Leijdekker'. A 'Running' button is visible. The main area has tabs for 'Implementation' (selected), 'Log', and 'History'. On the left, under 'API: SalesOrders', there's a 'POST /sync/{orderId}' endpoint. To its right, under 'Simple Mock Responses', is a box titled '200/OK' containing the text 'Success response'. On the far right, there are buttons for 'View and Test' (highlighted in blue), 'Copy URL', 'Publish to API Management', and a message 'No updates to push'.

- Test the mock app by filling out a value in the **orderId** field, and clicking on the **Try it out!** button:

This screenshot shows the configuration for the 'SalesOrders' API endpoint. The left sidebar lists resources and the current endpoint is 'POST /sync/{orderId}'. The main panel shows a 'Response Class (Status 200)' section with a 'Success response' example. Below it, the 'Parameters' table has one entry: 'orderId' with value '1234'. A 'Try it out!' button is shown next to the parameter input. Further down, there's a 'Curl' section with a command line, a 'Request URL' input field with the value 'https://64286c3e561fc85707011eabb-integration.cloud.tibcoapps.com:443/xrzzwukx3lcyazikn445ehjzcsizvd/sync/1234', and a 'Response Body' preview showing the JSON response: '{ "_response_string": "Order has been created with SalesDocumentID 00000013572" }'.

2.3 Import API Specs

2.3.1 Getting Ready

In this lab you'll import additional API specifications. In order to do this:

1. Navigate to the API specifications by clicking on the **API Specs** menu item.
2. Select the group you've created the **SalesOrders** API specification in, e.g. **MyTCIWorkshop**.
3. Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'Oregon' and other icons. The main menu has tabs for 'Apps', 'API Specs' (which is selected), 'Connections', 'Extensions', 'VPN Connections', and 'Downloads'. On the left, a sidebar titled 'All API specs' shows a 'GROUP' section with 'Default' and 'MyTCIWorkshop' selected. The main content area is titled 'MyTCIWorkshop' and lists one API specification: 'SalesOrders' (Version 1.0, Last modified 08 May 2019 07:41 pm Central European Summer Time). A search bar and a 'Create' button are at the top right of the list table.

2.3.2 How to Do It

1. Click on the **Import** button to import the API specifications:

This screenshot is identical to the previous one, but the 'Import' button in the top right corner of the 'Create' button's row is highlighted with a red circle.

2. Select **Import from filesystem**,

This screenshot shows the same interface as before, but a dropdown menu has appeared over the 'Import' button. The menu items are 'Import from filesystem' (which is highlighted in blue), 'Import from URL', and 'Import from Github'.

and select the following files from the \API Specs directory:

- **SAPHanaPurchaseOrders.json**
- **SAPOrders.json**
- **sfContacts.json**

3. Once these files have been uploaded, your screen should look similar to this:

The screenshot shows the TIBCO CLOUD Integration interface. The top navigation bar includes 'TIBCO CLOUD Integration', 'Oregon' location, and various icons. Below the navigation is a sub-menu with 'All API specs', 'GROUP', and a '+' icon. The main content area is titled 'MyTCIWorkshop' and lists four APIs: SalesOrders, SAPHanaPurchaseOrders, SAPOrders, and sfContacts, all grouped under 'Default'. A search bar and a 'Create' button are visible on the right.

API name	Version	Description	Last modified
SalesOrders	1.0		08 May 2019 07:41 pm Central European Summer Time
SAPHanaPurchaseOrders	1.0		08 May 2019 08:04 pm Central European Summer Time
SAPOrders	1.0		08 May 2019 08:04 pm Central European Summer Time
sfContacts	1.0		08 May 2019 08:04 pm Central European Summer Time

- For the SAPHanaPurchaseOrders we also create a Mock application as we did in 2.2.2.

- Hover over the **SAPHanaPurchaseOrders** record to the right of the preview icon, and select **Create Mock app** from the menu:

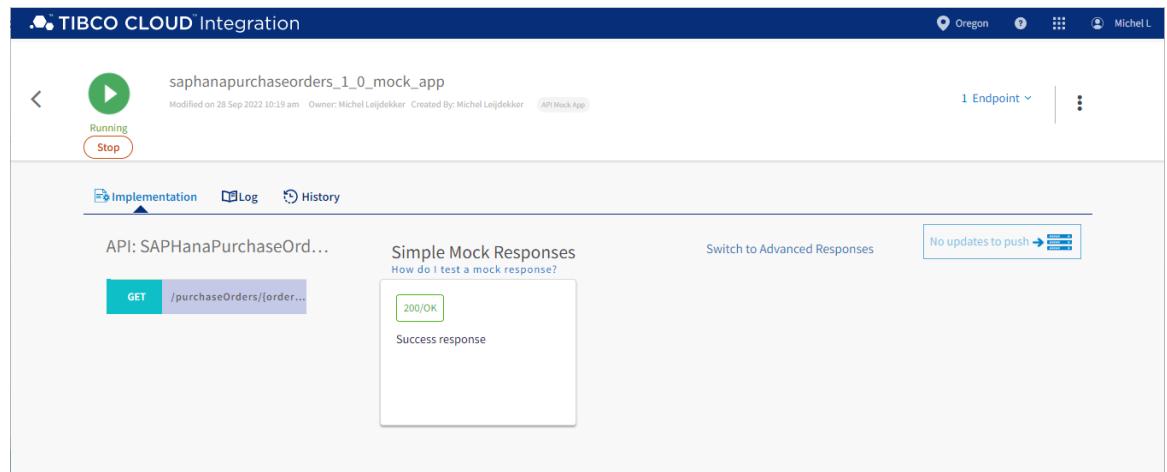
This screenshot shows the same interface as above, but with a context menu open over the SAPHanaPurchaseOrders row. The menu includes options like 'Create Mock app', 'Generate Node.js code', 'Clone', 'Edit', and 'Remove'.

API name	Version	Description	Group	Last modified
SalesOrders	1.0		MyTCIWorkshop	23 Sep 2022 7:28 pm
SAPHanaPurchaseOrders	1.0		MyTCIWorkshop	23 Sep 2022 7:57 pm
SAPOrders	1.0		MyTCIWorkshop	23 Sep 2022 7:57 pm
sfContacts	1.0		MyTCIWorkshop	23 Sep 2022 7:57 pm
TIBCO Cloud Integration PetStore Sam...	1.1	To get you started with the API Modeling and Mock capabilities of TIBCO Cloud Int...	Default	21 Sep 2022 1:57 pm

- Create a mock app by clicking on the **Create** button in next form:

The dialog box is titled 'Create Mock app'. It contains a text input field labeled 'Give your new Mock app a name' with the value 'saphanapurchaseorders_1_0_mock_app'. At the bottom are 'Cancel' and 'Create' buttons.

- Click on the **Simple Mock Responses** Tile to specify the response message that the mock service will return.



In the Edit Mock response wizard, copy and paste the following json snippet

```
{
  "d": {
    "CreationDate": "/Date(1472774400000)/",
    "CreatedByUser": "CB9980000027",
    "DistributionChannel": "10",
    "OverallSDProcessStatus": "20",
    "OverallTotalDeliveryStatus": "B",
    "SalesDistrict": " ",
    "SoldToParty": "17100001",
    "TotalNetAmount": "353.50",
    "TransactionCurrency": "USD",
    "SalesOrderType": "OR",
    "CustomerPurchaseOrderDate": "/Date(1472774400000)/",
    "OverallSDDocumentRejectionSts": "A",
    "ShippingCondition": "01",
    "IncotermsTransferLocation": "Palo Alto",
    "IncotermsVersion": " ",
    "SalesOrderDate": "/Date(1472774400000)/",
    "HeaderBillingBlockReason": " ",
    "SalesGroup": " ",
    "SalesOrder": "2",
    "ShippingType": " ",
    "IncotermsLocation2": " ",
    "DeliveryBlockReason": " ",
    "IncotermsLocation1": "Palo Alto",
    "AssignmentReference": " ",
    "OrganizationDivision": "00",
    "PurchaseOrderByCustomer": "John::Fisher::jfisher@acme.com",
    "SalesOrganization": "1710"
  }
}
```

```

        "IncotermsClassification": "123",
        "SalesOffice": " ",
        "TotalCreditCheckStatus": " ",
        "CustomerPurchaseOrderType": "123",
        "PricingDate": "/Date(1472774400000)/",
        "CustomerPaymentTerms": "0004",
        "LastChangeDate": "/Date(1472774400000)/",
        "SDDocumentReason": " ",
        "RequestedDeliveryDate": "/Date(1472774400000)/",
        "PaymentMethod": " ",
        "LastChangeDateTime": "/Date(1472774400000)/"
    }
}

```

- Click “**Save**”

200/OK

```

1 {
2   "d": {
3     "CreationDate": "/Date(1472774400000)/",
4     "CreatedByUser": "CB9980000027",
5     "DistributionChannel": "10",
6     "OverallSDPProcessStatus": "20",
7     "OverallTotalDeliveryStatus": "B",
8     "SalesDistrict": " ",
9     "SoldToParty": "17100001",
10    "TotalNetAmount": "353,50".
11

```

Generate response from schema Cancel Save

- After the save action click **Update Mock app** in the top right corner.
- Once the mock app is running again, hover over the **Endpoint** link, and select **View and Test** from the menu:

TIBCO CLOUD Integration

saphanapurchaseorders_1_0_mock_app

Running

Implementation Log History

Simple Mock Responses

How do I test a mock response?

GET /purchaseOrders/{order...}

200/OK

Success response

View and Test

Copy URL

Publish to API Management

No updates to push

- Verify if the mock service response is as expected.

- If everything is fine, we can change the endpoint visibility to refer to the Tibco Cloud Mesh. This will make sure the endpoint is only available as a private endpoint. To do this, click the < icon to navigate back to the API overview screen. From the menu on the top right corner, select the option **“Set endpoints private to my Tibco Cloud”** and click **Update**.

2.4 Additional Reading

- [How to Model Your API with TIBCO Cloud Integration](#)
- [TIBCO Cloud Integration - API Modeler](#)
- [My first API with API Modeler](#)
- [TIBCO Cloud Integration - API Mock App](#)
- [Testing APIs with Mock apps](#)
- [Meter Data Service API Mock for TIBCO Cloud Integration](#)

3. Build and Deploy Your API

In this section you have two options:

1. You will import the exported Business Works applications into the TIBCO Cloud Business Works runtime environment.
2. You'll push an app implemented in BusinessWorks from Business Studio to TIBCO Cloud. The lab instructions for this part has been added as an additional lab, since it will require the install of a local Business Works Studio environment.

Functionally, the app implements a system API that gets purchase orders from SAP Hana.

After that you'll create a Flogo app that implements a process API that creates sales orders, by first invoking the system API that gets a purchase order from SAP Hana, then invoking a system API that gets contact details from Salesforce, and finally creates a sales order by invoking a system API for SAP ERP.

- BusinessWorks Apps in TIBCO Cloud Integration
 - Option 1: Import the Business Works Application via EAR import.
 - Option 2: Connect Business Studio to TIBCO Cloud
 - Import a Project in to Business Studio
 - Push the Project to TIBCO Cloud and Test It
- Flogo Apps in TIBCO Cloud Integration
 - Create the Skeleton Flogo App from the API Specification
 - Implement the Flogo App
 - Push the Flogo App to TIBCO Cloud and Test It

3.1 BusinessWorks Apps in TIBCO Cloud Integration

3.1.1 How to Do it: Import the Business Works Application via EAR import

1. We will import the EAR export files and the corresponding manifest files into the Tibco Cloud Integration.

- In the Integration Apps, click the Create/Import button

The screenshot shows the TIBCO CLOUD Integration dashboard. The top navigation bar includes 'TIBCO CLOUD' and 'Integration'. Below it, there are tabs for 'Apps', 'Marketplace', 'Connections', and 'Environment & Tools'. The main area is titled 'Apps' and contains a search bar with 'Search app name' and a magnifying glass icon. In the top right corner of the main area, there is a blue button labeled '+ Create/Import'.

- Then select the "All app types" menu option from the wizard and select the "Import a BusinessWorks app"

This screenshot shows the 'What do you want to build?' wizard. On the left, there's a sidebar with filters for Category (All), Deployment (TIBCO Cloud), App Owner (All apps in org), App Status (All), and Tags (0/0). The main panel has a title 'What do you want to build?'. It lists several categories: 'Quickstart' (Get started, All app types), 'Application Integration' (API-based Integration, Messaging/Event-based Integration, Event Processing, IoT & Edge Processing, Mobile/Web Event-driven Integration), 'Data Integration' (Legacy Interface Integration, File-based Integration, Data Replication, Data synchronization and migration, Standard Protocol-based B2B), 'Process Automation' (Business Automations), and 'APIs' (Create an app from OpenAPI, Create an app from GraphQL). A callout highlights the 'Import a BusinessWorks app' option under the 'All app types' section.

This screenshot shows the 'Import a BusinessWorks app' wizard. The left sidebar is identical to the previous screenshot. The main panel has a title 'Import a BusinessWorks app'. It displays a summary: 'BusinessWorks™ Deploy a TIBCO BusinessWorks app on TIBCO Cloud. Use the BusinessWorks app to integrate enterprise apps and orchestrate web services across hybrid environments.' Below this is a dashed rectangular area with two icons: a .ear file and a .json file. A blue button at the bottom right says 'Import BusinessWorks app'. A callout highlights this button.

4. Select the following EAR- and manifest files.

Service	Files
Contact Service	SalesForceAPI_1.0.0.ear + manifest.json
Orders Service	SAPOrdersAPI_1.0.0.ear + manifest.json
Sales Order Service	SAPHanaAPI_1.0.0.ear + manifest.json

5. After successful installation of the 3 Business Works applications, your screen should look similar to the following:

Important Note: The trial subscription only allows to run two apps at the same time. Therefor we will use API Mocks to simulate 2 of the 3 required Business Works solutions.

3.2 Flogo Apps in TIBCO Cloud Integration

In this lab you'll implement a process API that creates sales orders, by first invoking the system API that gets a purchase order from SAP Hana, then invoking a system API that gets contact details from Salesforce, and finally creates a sales order by invoking a system API for SAP ERP.

You will first create a "skeleton" Flogo app from the **SalesOrders** API specification. Then you will implement the flow with the 3 system API calls, to finally push the Flogo app and test it.

3.2.1 Getting Ready

To be able to create the "skeleton" Flogo app from the **SalesOrders** API specification, do the following:

1. Navigate to the Integration Apps specifications by clicking on the **Integration** hexacon in the main menu.
2. Your screen should look similar to:

The screenshot shows the TIBCO CLOUD Integration Apps interface. On the left, there is a sidebar with filters for Category (All), Deployment (All), App Owner (All apps in org), App Status (All), and Tags. The main area displays a table of apps with columns: Category, Name, Deployment, Last Modified, Last Started, Instances, and Status. There are three entries:

Category	Name	Deployment	Last Modified	Last Started	Instances	Status
Integrate	SAPOrdersAPI	TIBCO Cloud	24 Sep 2022 10:29 am	23 Sep 2022 2:30 pm	1	Running
Integrate	SalesForceStub	TIBCO Cloud	23 Sep 2022 1:43 pm	23 Sep 2022 1:25 pm	1	Running
Integrate	SAPHana_SalesOrder_Stub	TIBCO Cloud	23 Sep 2022 2:00 pm		0	Stopped

At the bottom right of the main area is a blue arrow icon pointing right.

3.2.2 How to Do It: Create the Skeleton Flogo App from the API Specification

To create a "skeleton" Flogo app from on the **SalesOrders** API specification, do the following:

1. Click on the blue Create/Import button, this will open the following wizard

- Select the API based Integration option from the left menu and select your SalesOrders API specification.

The screenshot shows the TIBCO Cloud Integration platform. On the left, there's a sidebar with filters for 'Category', 'Deployment', 'App Owner', 'App Status', and 'Tags'. The main area shows a 'Quickstart' section with 'Get started' and 'All app types'. Below that is the 'Application Integration' section, which is expanded to show 'API-based Integration' (selected), 'Messaging/Event-based Integration', 'Event Processing', 'IoT & Edge Processing', 'Mobile/Web Event-driven Integration', 'Data Integration', and 'Process Automation'. A modal window titled 'What do you want to build?' is overlaid. It has a 'Flogo' logo and instructions: 'Create an app from an OpenAPI Specification using an existing file stored in API Modeler or by uploading an Open API Specification file.' It includes tabs for 'API specs' (selected) and 'Upload a file'. Below is a table of API specifications:

API Name	Version	Group	Last Modified
TIBCO Cloud Integration PetStore Sample	1.1	Default	21 September 2022 1:57 PM
SalesOrders	1.0	MyTCIWorkshop	23 September 2022 7:28 PM
SAPHanaPurchaseOrders	1.0	MyTCIWorkshop	23 September 2022 7:57 PM
SAPOrders	1.0	MyTCIWorkshop	23 September 2022 7:57 PM
sfContacts	1.0	MyTCIWorkshop	23 September 2022 7:57 PM
MichelTest	V1	MLE_Project1	22 September 2022 1:34 PM

Import OpenAPI spec

- Click on the **Import OpenAPI spec** button, to generate the skeleton. After generation, your screen should look like this.

The screenshot shows the 'Flows' tab for the 'New_Flogo_App_0' integration. At the top, there's a header with the integration name, owner (Michel Leijdekker), version (v. 1.0.0), and deployment status ('Not deployed'). Below the header, there are tabs for 'Flows', 'Endpoints', 'Monitoring', 'Environment controls', 'Logs', 'History', and 'Execution History'. Under the 'Flows' tab, there's a 'Create' button and a 'Trigger View' dropdown. A single flow is listed: 'postSync_orderid_POST' with a 'ReceiveHTTP' trigger. To the right of the flow list are buttons for 'Properties', 'Schemas', 'Validate', and 'Push'.

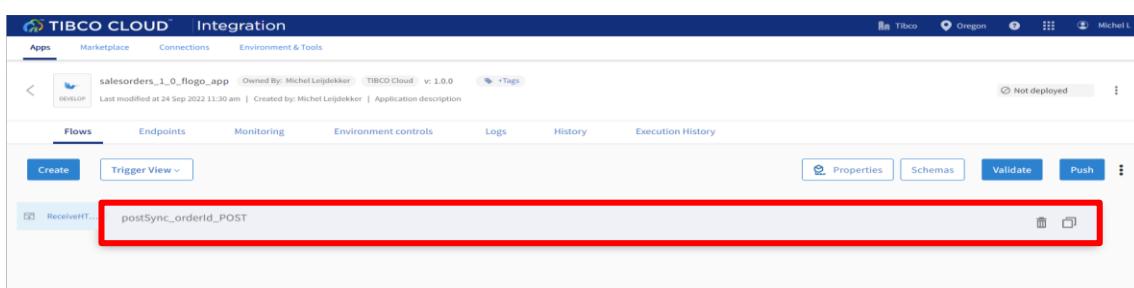
- Click on the "New_Flogo_App_0" name of the integration and rename it to "**salesorders_1_0_flogo_app**".

3.2.3 How to Do It: Implement the Flogo App

In this lab, you'll create a process API that creates sales orders by making 3 system API calls. The address of first system API call is the URL of the BusinessWorks app you have pushed and tested in the previous lab. The addresses of the other API calls are from apps we have already deployed, and their URLs are specified below.

To implement the process API, do the following:

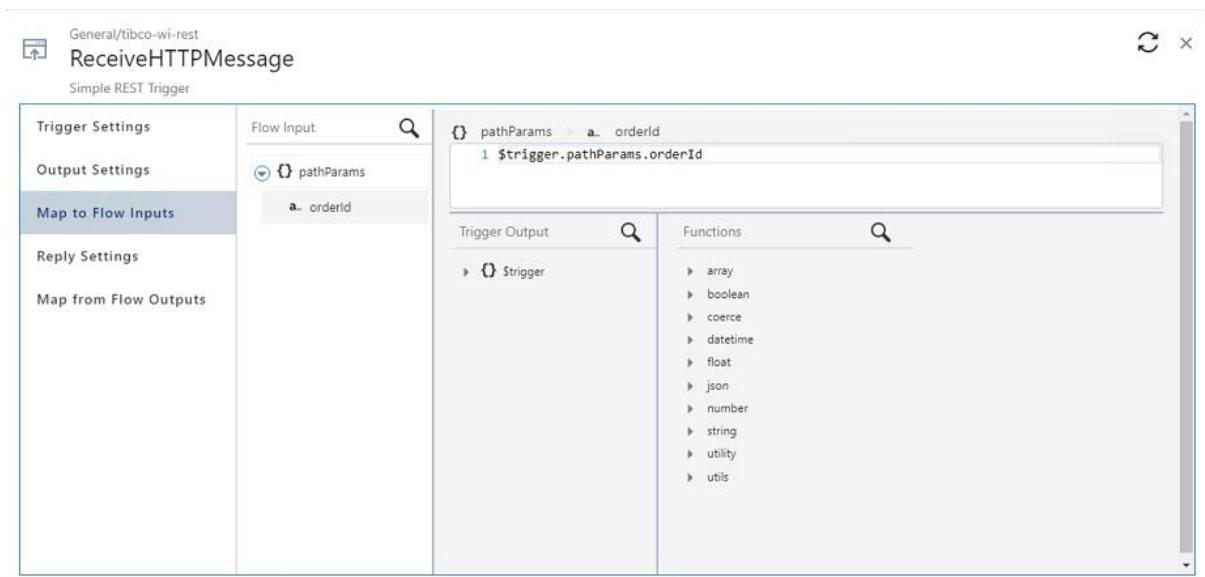
1. Navigate from the app **salesorders_1_0_flogo_app** to the skeleton flow **postSync_orderId_POST**.



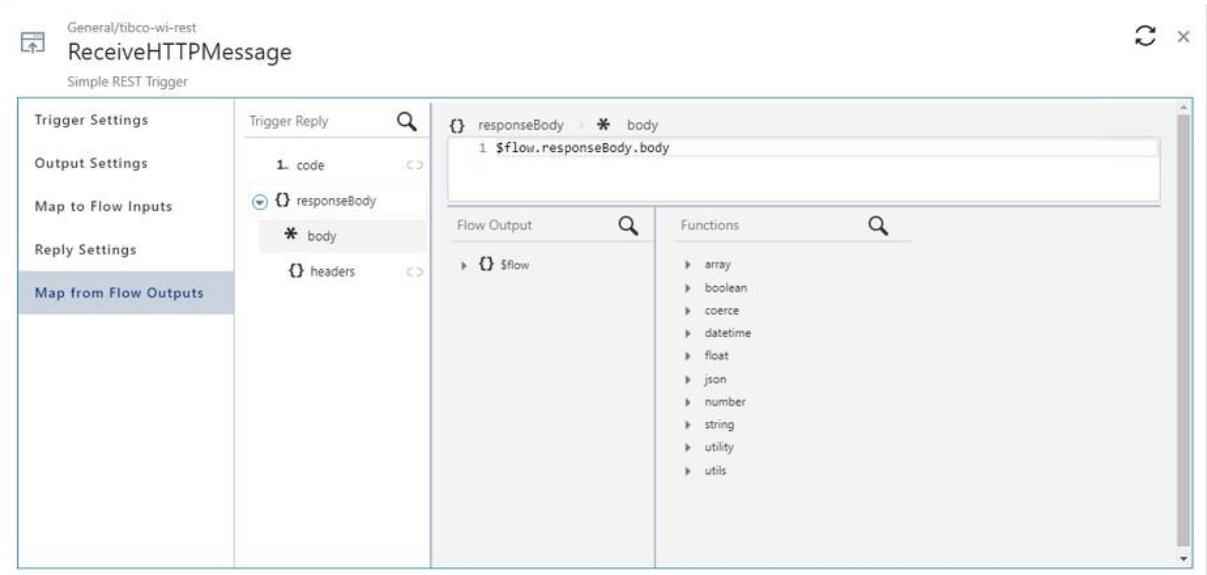
2. To enable data to flow between the trigger and the flow, you first need to map the trigger output to the flow input, and flow data to the trigger reply.

Click on the icon to open the configuration of the **ReceiveHTTPMessage** trigger.

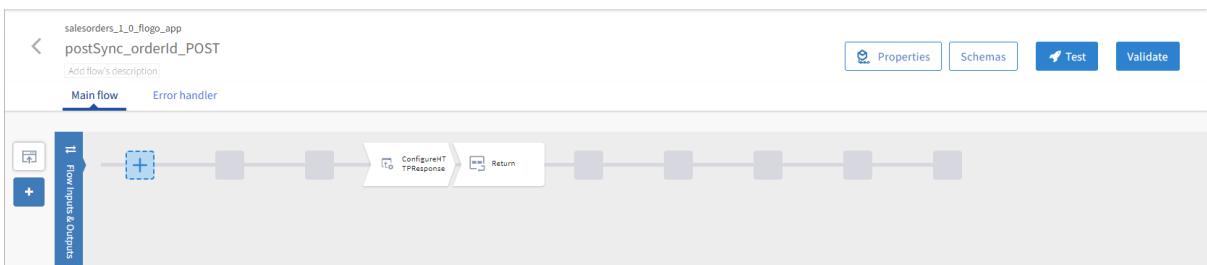
3. Check the auto mapped mappings from the trigger output to the flow input as follows:



- Also check the mapping of the flow data to the trigger reply as follows:

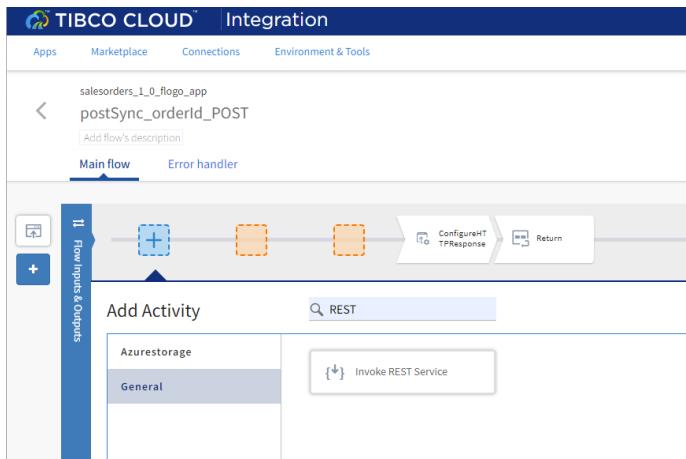


- Close the trigger configuration screen, and move the **ConfigureHTTPResponse** and **Return** tiles at least 3 positions to the right.



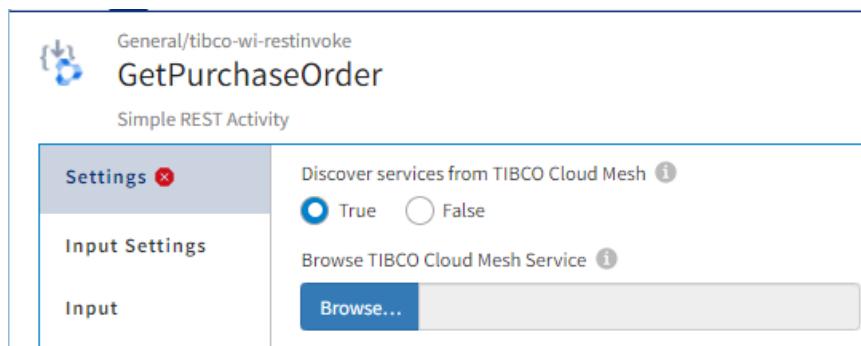
Creating the Activities

- Create a first activity, by clicking on the left-most + and navigate through **General > Invoke REST Service**.

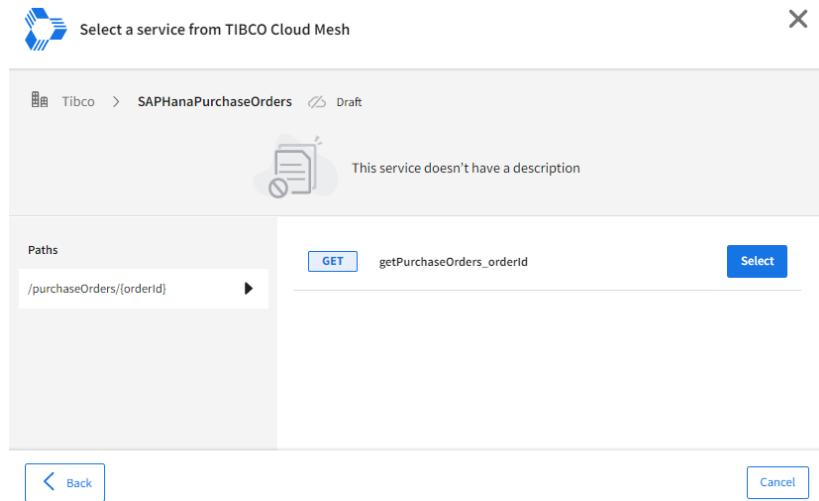


7. Configure this activity as follows:

- Give it a name: **GetPurchaseOrder**.
- The SalesOrder Service that connects to SAP Hana, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**

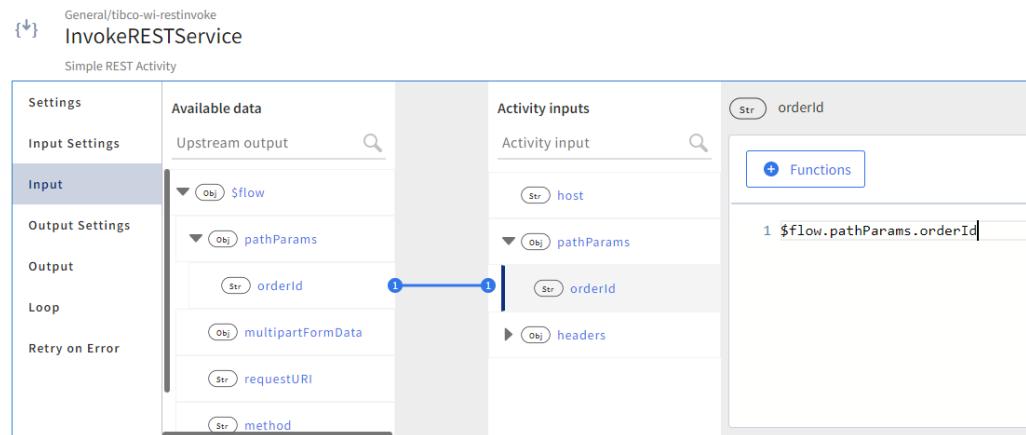


- Click on the **SAPHanaPurchaseOrders** API Definition from the list. In the next wizard screen **select** the operation.

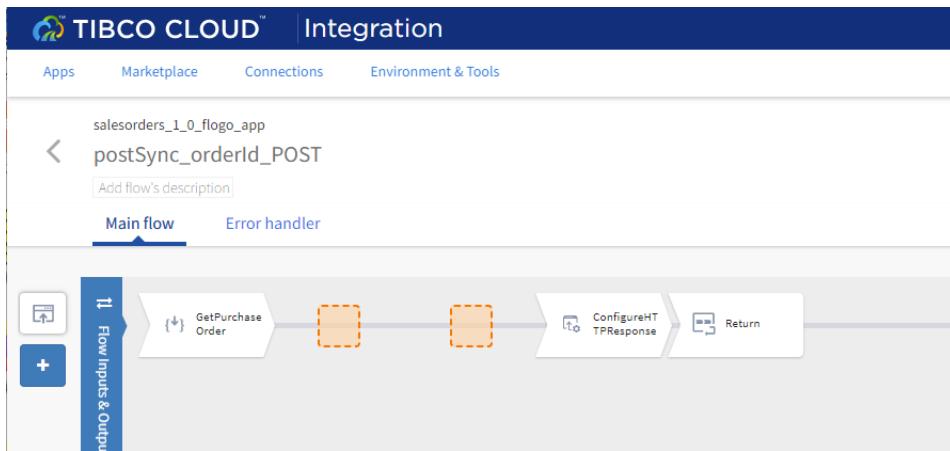


- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
<u>pathParams.orderId \$flow.pathParams.orderId</u>	

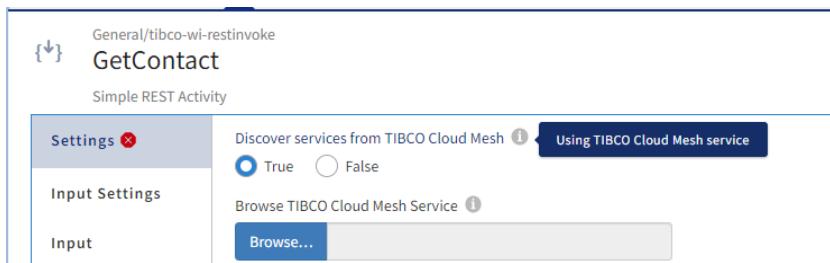


- Close the activity configuration screen, the integration should now look like the picture below.

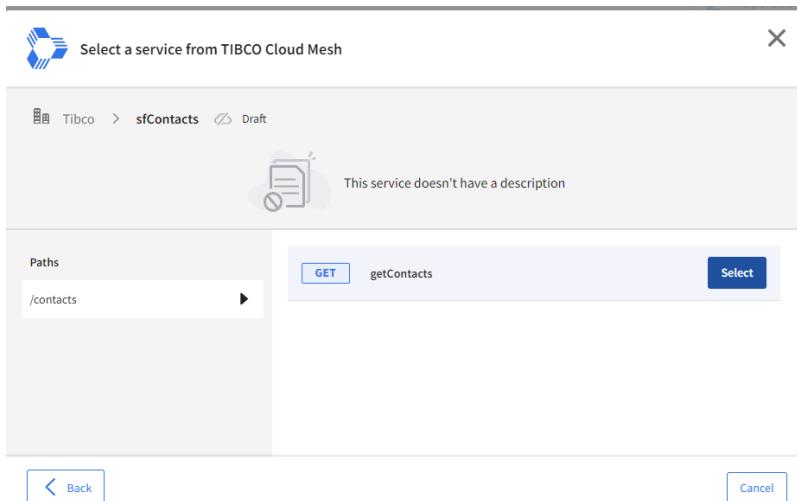


9. Now create a second activity, by clicking on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

- Give it a name: **GetContact**.
- The Contact Service that connects to SalesForce, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**



- Click on the **sfContacts** API Definition from the list. In the next wizard screen select the operation.

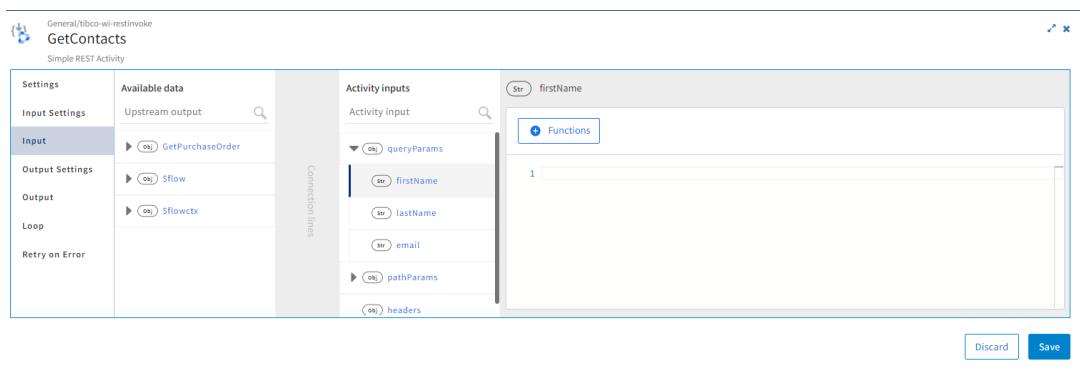


- In the **Input Settings** section, check the imported query parameters.
- In the **Input** section, create the mapping.

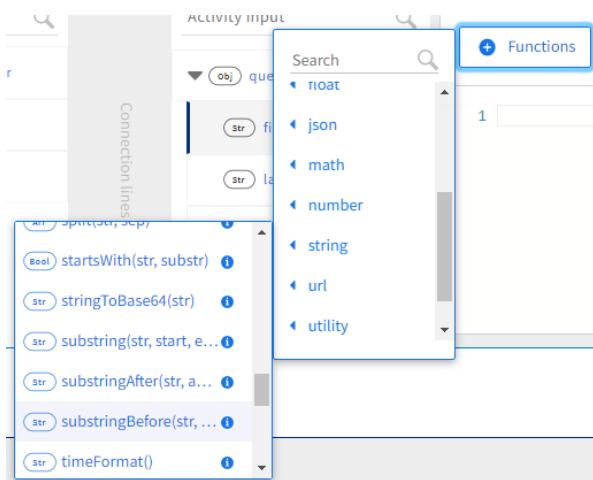
Activity Input	Upstream Output
<code>queryParams</code>	<code>string.substringBefore(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":")</code>
<code>queryParams</code>	<code>string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")</code>
<code>queryParams</code>	<code>string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")</code>
<code>.email</code>	

- The first mapping we will create manually using the Function wizard.

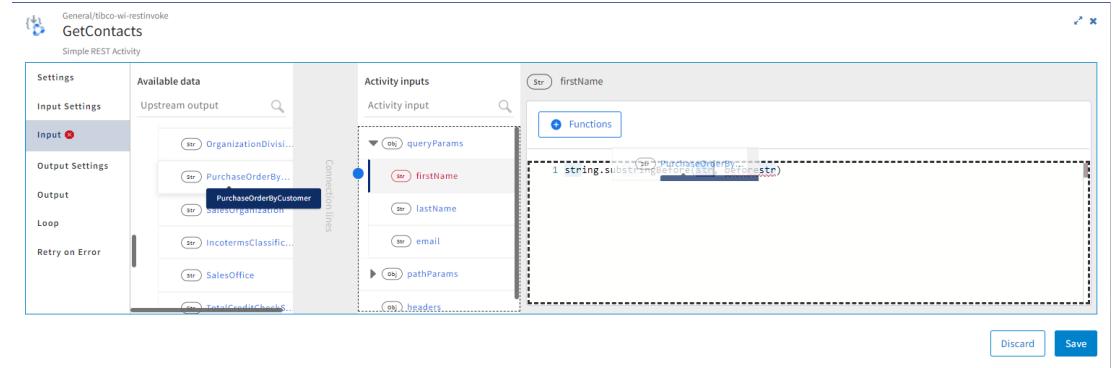
1. Select the *Activity Input : queryParams.firstName* and click on the Functions button.



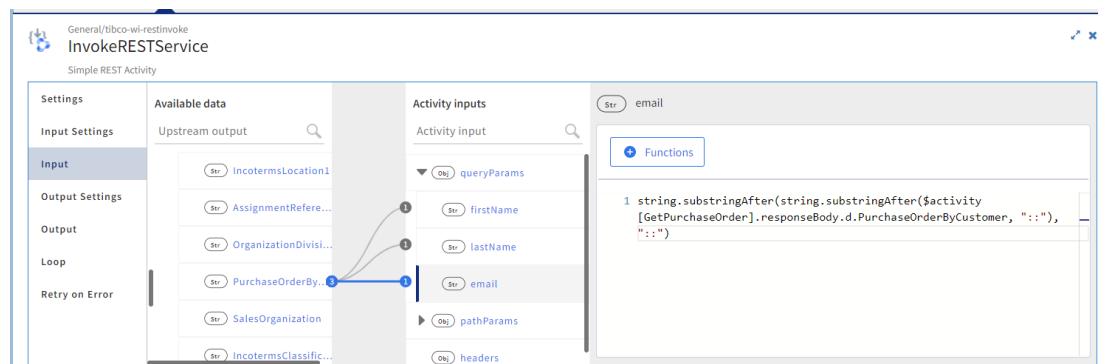
2. In the Functions dropdown Search in the string functions for the **substringBefore** function and select this to drop it to the mapping area.



- Set the first string **str** parameter to the PurchaseOrderByCustomer field from the responsebody of the GetPurchaseOrder Activity.



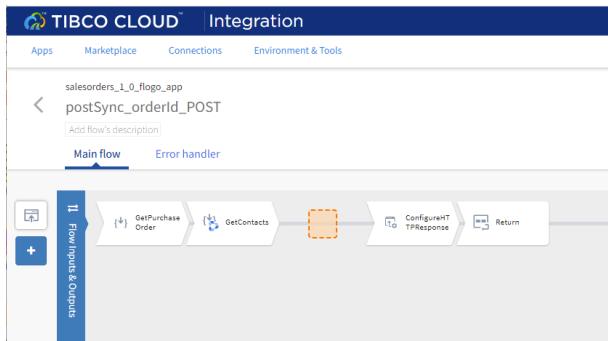
- Set the second string **beforestr** to ":", by typing into the mapping.



- Now create the other two mappings for the other two fields, by copy and pasting from this table.

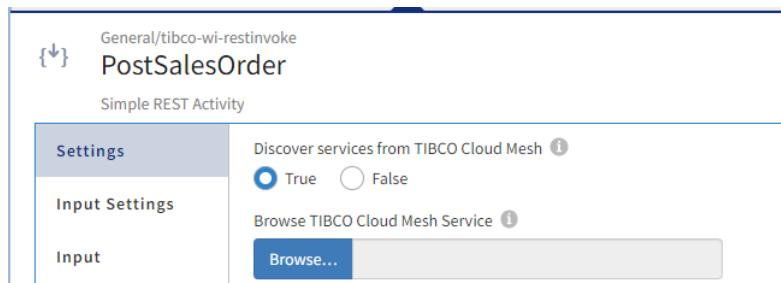
Activity Input	Upstream Output
queryParams	string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
.lastName	string.substringBefore(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
queryParams	string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")
.email	string.substringAfter(string.substringAfter(\$activity[GetPurchaseOrder].responseCodes["200"].d.PurchaseOrderByCustomer, ":"), ":")

- Close the activity configuration screen, the integration flow now should look like this

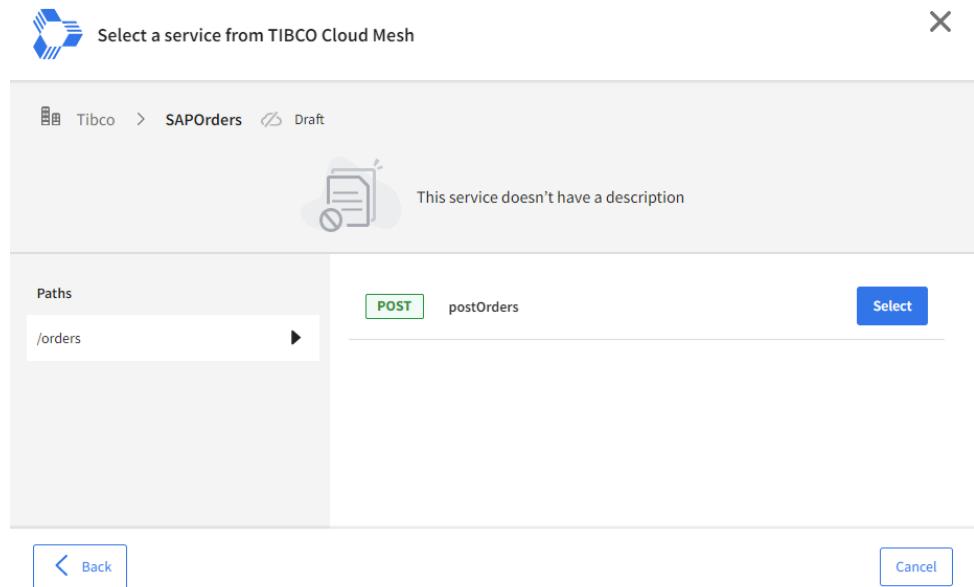


11. Now we create the third activity, which corresponds to the SAP Order Creation Service. Click on the next + and navigate through **General > Invoke REST Service**. Configure this activity as follows:

- Give it a name: **PostSalesOrder**.
- The Sales Order Create Service that connects to SAP, we will retrieve from the Tibco Cloud Mesh. Select the **True** button for Discovering Services from the Tibco Cloud Mesh. Click **Browse..**



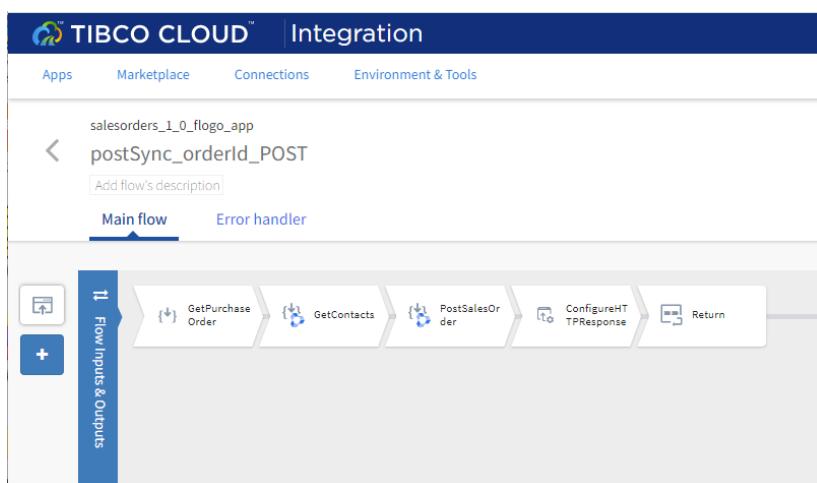
- Now select the SAPOrders from the list and select the postOrders Operation.



- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
body.Email	\$activity[GetContacts].responseCodes["200"].email
body.firstName	\$activity[GetContacts].responseCodes["200"].firstName
body.item	"1"
body.lastName	\$activity[GetContacts].responseCodes["200"].lastName
body.phone	\$activity[GetContacts].responseCodes["200"].mobile
body.price	\$activity[GetPurchaseOrder].responseCodes["200"].d.TotalNetAmount

12. Close the activity configuration screen. The flow now looks similar to the following picture.



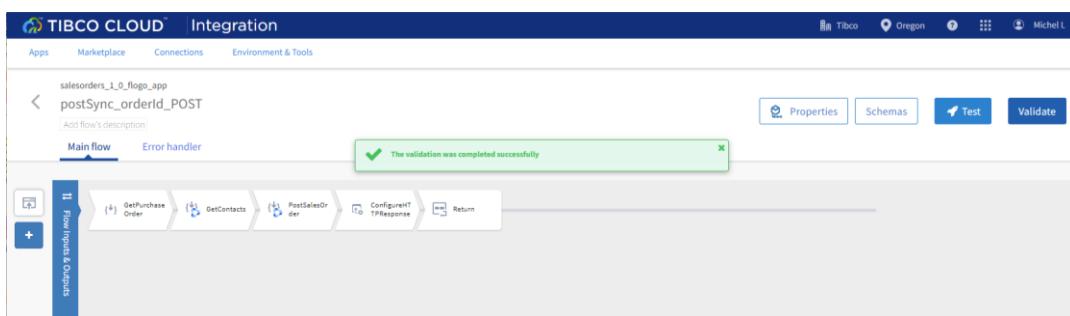
13. Now configure the **ConfigureHTTPResponse** activity as follows:

- In the **Input** section, create the following mapping:

Activity Input	Upstream Output
code	200
Input.body._response_string	\$activity[PostSalesOrder].responseCodes["200"]._response_string

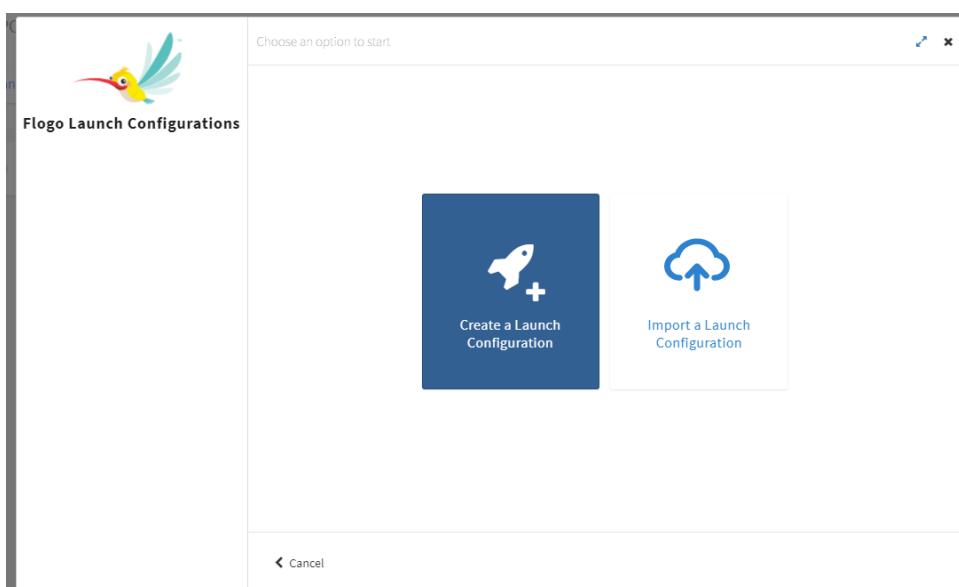
14. Close the activity configuration screen.

15. Now Validate the Integration, the result should look something like:

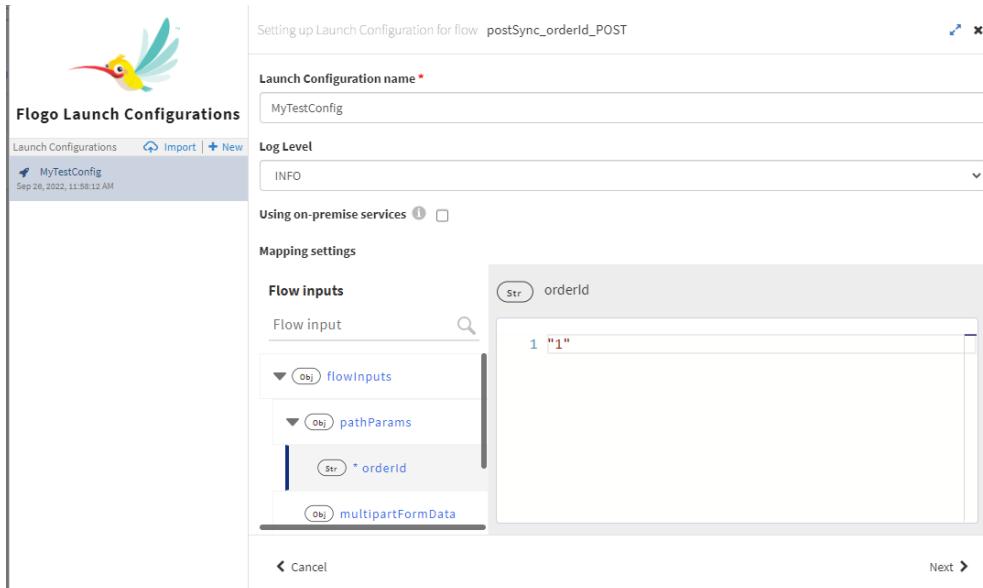


3.2.4 Push the Integration App to TIBCO Cloud and Test It

1. Click on the Blue **Test** Button to start the Test configuration wizard. First you need to create a Launch Configuration for this test.



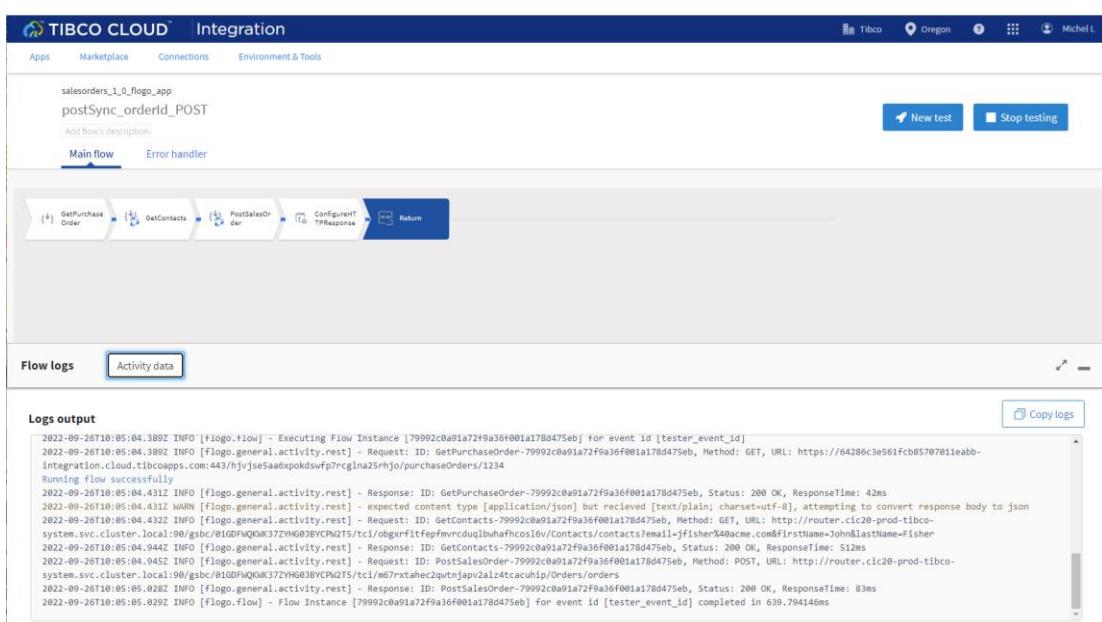
- Click on the corresponding tile and give your configuration a name. Create a mapping for the integration input parameter **orderId** as shown below.



- Click the **Next** button in the right lower corner and check the summary. Now click the Run button in the bottom right corner to start the test run.



- Now the integration flow test log will open and show the status of the test run.



5. You can introspect individual service and API calls on input and output result, by just clicking on the flow diagram and select the activity you want to inspect.

The screenshot shows the TIBCO CLOUD Integration interface. At the top, there's a navigation bar with 'TIBCO CLOUD' and 'Integration'. Below it, a sub-navigation bar has 'Main flow' selected. The main area displays a flow diagram with activities: 'GetPurchaseOrder', 'GetContacts', 'PostSalesOrder', 'ConfigureTTPResponse', and 'Return'. Below the flow diagram, there are two tabs: 'Activity data' (selected) and 'Flow logs'. Under 'Activity data', for the 'GetContacts' activity, there are sections for 'Inputs' and 'Outputs'. The 'Inputs' section shows a JSON object with fields like 'email', 'firstName', and 'lastName'. The 'Outputs' section shows a JSON object with fields like 'statusCode', 'responseCodes', and 'mailingStreet'.

To push the Integration app to the runtime environment and test it, do the following:

1. Click the **Push app** button.

The screenshot shows the deployment interface. At the top, there's a header with 'Tibco' and 'Oregon'. Below it, there's a status bar showing '0 Instances' and 'Deploying'. At the bottom, there are buttons for 'Properties', 'Schemas', 'Validate', 'Push' (which is highlighted in blue), and a more options menu.

2. Now we have to scale the application to start an instance of the integration

The screenshot shows the application management interface. It displays basic information about the app: 'salesorders_1_0_flogo_app', 'Owned By: Michel Leijdekker', 'TIBCO Cloud v: 1.0.0', and 'Last modified at 26 Sep 2022 11:48 am | Created by: Michel Leijdekker | Application description'. At the bottom right, there's a 'Scale' button with the value '1' and a status indicator 'Stopped'.

3. Once the **salesorders_1_0_flogo_app** app is running, click on the app and select the **Endpoint** tab. Select **Test** from the possible actions

- Click on the green POST operation to open the API details.

- Test the **salesorders_1_0_flogo_app** app by clicking on the **Try it out!** button. Fill out a value in the **orderId** field, and click on the blue Execute bar. Now watch the result.

The screenshot shows the Tibco Business Works API endpoint configuration interface. At the top, there are tabs for Endpoints, Monitoring, Environment controls, Logs, History, and Execution History. The Endpoints tab is selected. Below the tabs, there's a 'Parameters' section with a table. A row for 'orderid' is highlighted with a red asterisk indicating it's required. The 'orderid' column shows 'string (path)' and the 'value' column contains '1'. Below the table are 'Execute' and 'Clear' buttons. Under the 'Responses' section, there's a 'Curl' block with a command, a 'Request URL' block with a URL, and a 'Server response' block with a JSON response body. The response body is a single object with a key '_response_string' and a value 'Sales Order has been created with SalesDocumentID 000013572'. There are also 'Download' and 'Copy' buttons next to the response body.

You now have successfully build the Sales Orders integration, that uses Tibco Business Works integrations, API definitions and Mock Applications.

3.3 Additional Reading

BusinessWorks (Cloud Integration)

- [TIBCO Cloud Integration: Getting Started with BusinessWorks Applications](#)
- [TIBCO Business Studio™ - Cloud Edition](#)
- [TIBCO ActiveMatrix BusinessWorks™ Plug-ins](#)
- [TIBCO® Cloud Integration Frequently Asked Questions](#)

Flogo (Cloud Integration)

- [TIBCO Cloud Integration: Getting Started with Flogo](#)
- [TIBCO Flogo® apps](#)
- [TIBCO Flogo® Connectors](#)

Project Flogo

- [Project Flogo™ Community Wiki](#)
- [Project Flogo](#)
- [Project Flogo Documentation](#)
- [Project Flogo Github Repos](#)

Extra LAB: Business Events Simple Decision Service

Simple Decision Services (SDS) is a capability of **TIBCO Cloud Events. (TCE)**

SDS is an easy and simplified approach to create stateless decision tables. It has browser-based authoring experience without any complexity.

Business (or non-technical) users can now easily define their business logic in the form of simple conditions and actions in an excel like format called as a "Decision Service". These services can be exposed as RESTful APIs for internal and external consumption.

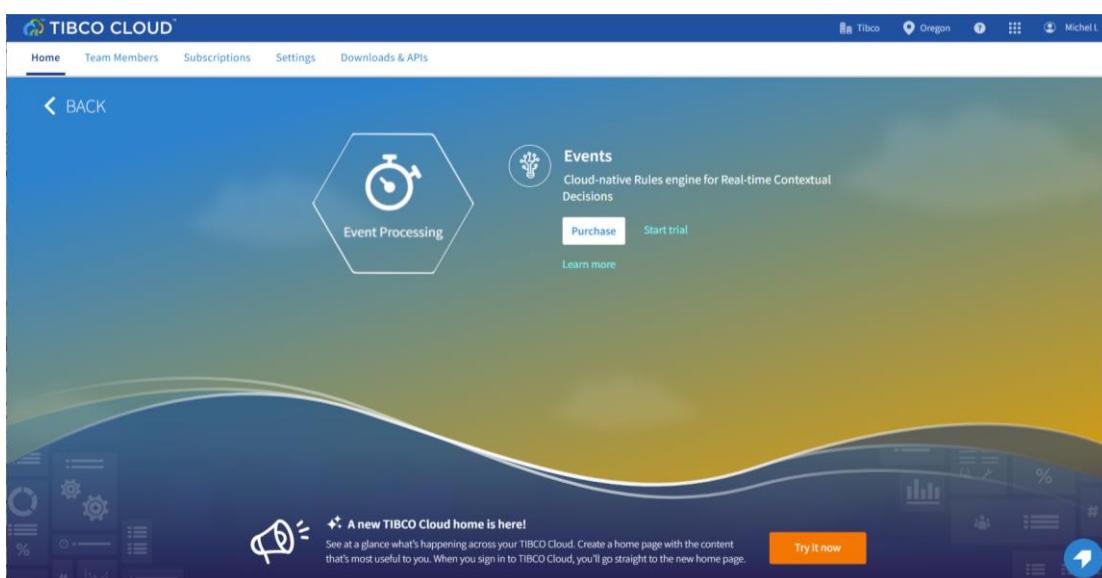
This walkthrough provides a step-by-step journey to create a Simple Decision Service. You can have one or more Simple Decision Services as part of a single TCE application.

What You Will Learn

- Understanding the concept of Simple Decision Service(s).
- How to create, deploy and test a Simple Decision Service(s).

Getting Ready

1. Extend your trial with a Business Events trial. In the Home screen, navigate to the Event Processing hexacon and Start trial.



- Follow the necessary registration steps. After registration, the Cloud Event environment opens in the window. Close this window and return to the Home menu. This should now show the Event Processing as opened.



Creating a Base TCE Project

- Login to TIBCO Cloud Events and select the Org you want to use.
- Click the Push App button (on the top right side).



Figure 1: Push App Button

- On clicking the Push App button, select the Decision Service tab in the wizard shown below.
- Select the desired Application Size, No.of Instances and the Name of the Application before pushing the application using the Push button.

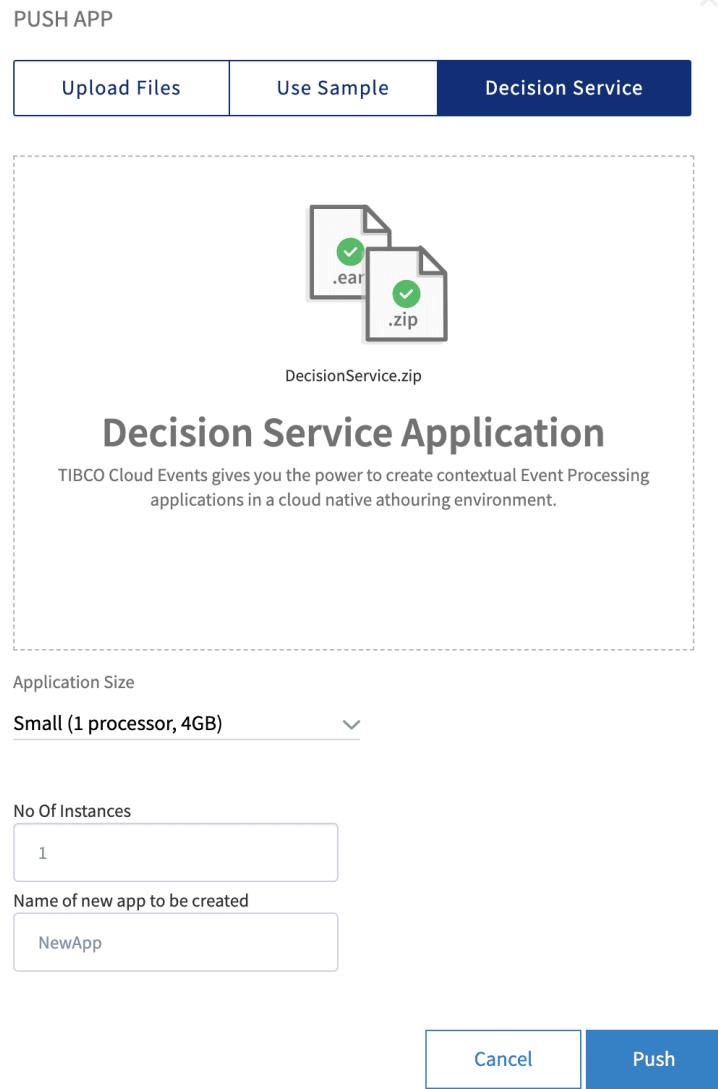


Figure 2: SDS - Push Application

- Once you Push the application, it takes a few seconds to Scale and come to the Running state. The Application itself is in the Draft stage scaled to the number of instances mentioned during deployment.

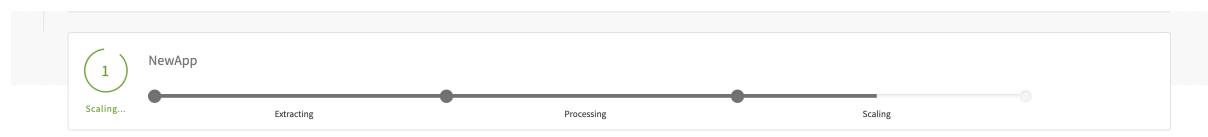


Figure 3: Application Push Stages



Figure 4: Application Deployment

- Now the base TCE project to create a SDS is available and deployed. Let's proceed ahead to implement a SDS now.

Implementing a Simple decision service

- Leverage TIBCO Cloud Events WebStudio to implement a Simple Decision Service.
- Click the WebStudio button (on the top right) to begin the implementation steps.

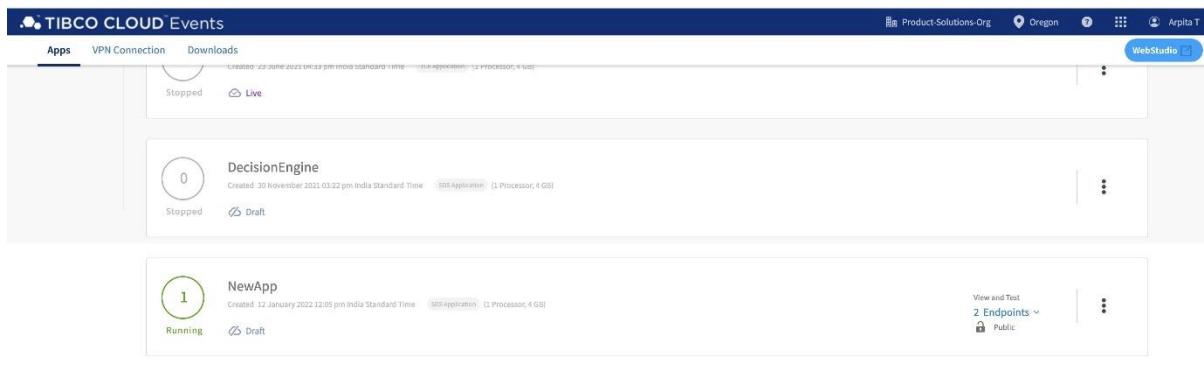


Figure 5: TIBCO Cloud Events Home - WebStudio option

- Within the WebStudio, click + sign on the Project Explorer tab (on the left) and then "Create a new decision service" option to add a new implementation of SDS.

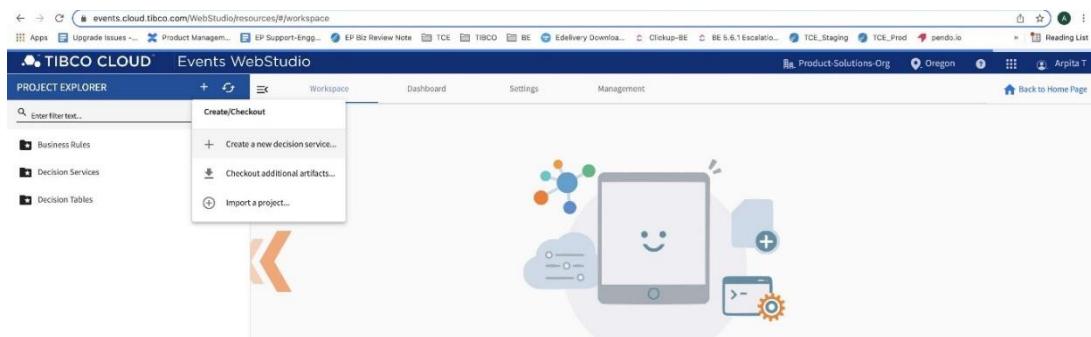


Figure 6: Create a new Decision Service option

4. Provide the required input details as prompted by the wizard. Project name from the dropdown list are the ones already deployed as part of the previous step. If the project is not previously checked out, it will be checked out during the SDS creation process.
5. Enter the input and output fields in the following wizards. The inputs and outputs are defined in the form of a simple name and data type format.

TIBCO CLOUD Events WebStudio

PROJECT EXPLORER + ⌂ Workspace Dashboard Settings Management

Product-Solutions-Org Oregon Arpita T Back to Home Page

DECISION SERVICE

Create new Decision Services

1 Decision Service Information **2 Decision Service Input** **3 Decision Service Output**

You are now on the first step toward creating your Decision Service! Enter a name for your service. By default, your service will be accessible through this service name, so it is helpful to pick a meaningful name here.

Project Name: NewApp

The selected project is not checked out, but will be automatically on track.

Decision Service Name: NewAppSDS

Service Description:

EXIT **NEXT** FINISH

Additional Details

You can change how your decision service will run here, or you can leave the default values and continue. These settings can always be changed later by editing the decision service.

Only output the first match

Effective Date/Time

Expiration Date/Time

Select a field to see additional details

Figure 7: Create Decision Service wizard

Workspace Dashboard Settings Management Back to Home Page

DECISION SERVICE

Create new Decision Services

1 Decision Service Information **2 Decision Service Input** **3 Decision Service Output**

Input Fields

Now let's define the input fields for our Decision Service. These fields can be used in your Decision Service Implementations to determine what the output of the service will be.

Field Name: Marks	Field Type: int	Description:	+ -
-------------------	-----------------	--------------	-------------------

EXIT BACK **NEXT** FINISH

Field Marks

Will this field contain a list of values? You can add the valid values here.

No value restrictions have been defined

Figure 8: Input fields of SDS

The screenshot shows the 'DECISION SERVICE' configuration page. At the top, there are tabs for 'Decision Service Information', 'Decision Service Input', and 'Decision Service Output'. The 'Decision Service Output' tab is active, indicated by a blue circle with a '3'. Below the tabs, a message says 'The final step is to define the output fields of your Decision Service'. A section titled 'Output Fields' contains a table with one row: 'Result' (Field Name), 'String' (Field Type), and an empty 'Description' field. To the right, a 'Field Result' panel asks if the field will contain a list of values, with a note that no value restrictions have been defined. Buttons at the bottom include 'EXIT', 'BACK', and 'FINISH'.

Figure 9: Output fields of SDS

6. Click Finish once the input and output fields are defined to create the default implementation of a SDS.
7. Click on Add row button to add the desired conditions (inputs) and actions (outputs).

The screenshot shows the 'NewAppSDS_defaultImpl' editor for a Decision Service Implementation. The title bar indicates the current artifact is 'NewApp > DecisionServices > NewAppSDS > NewAppSDS_defaultImpl'. A message at the top right says 'You are editing the latest version' and 'Artifact Type: Decision Service Implementation'. The main area is a 'Decision Table' with columns 'DT', 'CONDITION', and 'ACTION'. There are three rows: 1. ID: input.Marks (int), ACTION: Fail; 2. ID: >40, ACTION: Pass; 3. ID: =40, ACTION: Pass. On the right side, there are sections for 'Priority' (values 5, 5, 5), 'Columns', and 'Filters'. A toolbar at the top has buttons for 'SAVE', 'COMMIT', 'UNDO', 'REDO', and 'VALIDATE'.

Figure 10: Decision Service Implementation

Once the rows are defined, click Save button - this will enable the Commit option to commit the changes for an Approval process.

- ❖ Commit the changes with appropriate message for the Admin user to approve the changes. The changes need to be committed and approved before they can be deployed to the TCE application.

Deploying the Decision Service

1. After Commit action, the changes are available for Admin Approval process.

The screenshot shows the TIBCO Cloud Events WebStudio interface. At the top, there's a navigation bar with links like 'Events', 'WebStudio', 'Resources', and 'Dashboard'. Below the navigation is a header bar with sections for 'Recent User Activity', 'Project Snapshots', and 'Management'. The 'Recent User Activity' section shows a log entry from 'Tirodkar, Arpita' committing changes on 'Jan 13, 2022'. It includes a 'View Changes' button and 'Reject'/'Approve' buttons. The 'Project Snapshots' section displays two projects: 'NewApp' and 'NewTest'. 'NewApp' has 2 unresolved worklist items and 0 deployable artifacts. 'NewTest' has 0 unresolved worklist items and 3 deployable artifacts. Both projects show their last commit details and project statistics. There are also sections for 'AlertProcessing' and 'CreditCardApplication'.

Figure 11: Deployment Dashboard

2. Admin user can click on the eye button on the left side under the Recent User Activity log to Approve/Reject the committed changes.
3. Once the changes are approved, under the Project Snapshots section, left click on the Option menu (3 dots icon on the right side specific to your Project) and select "Generate Deployable" as shown below.

The screenshot shows the TIBCO CLOUD Events WebStudio dashboard. At the top, there are links for Apps, Upgrade Issues, Product Management, EP Biz Review Note, TCE, TIBCO, BE, Edeivery Download, Clickup-BE, BE 6.6.1 Escalation, TCE_Staging, TCE_Prod, and pendo.io. The dashboard has tabs for Workspace, Dashboard (selected), Settings, and Management. A search bar asks "Welcome back, Arpit! What would you like to do today?". On the right, there are counts for total unresolved item(s) (2) and total deployable artifact(s) (15). Below the search bar, there are sections for Recent User Activity and Project Snapshots.

- Recent User Activity:**
 - Approved [by Tirodkar, Arpit] (ID: 10022): New row added by Tirodkar, Arpit committed 30 minutes ago.
 - Tirodkar, Arpit Modified /DecisionServices/NewAppSDS/NewAppSDS_defaultImpl at 2:04 PM
 - Tirodkar, Arpit Committed /DecisionServices/NewAppSDS/NewAppSDS_defaultImpl at 2:04 PM
- Project Snapshots:**
 - NewApp**: Last commit by: Tirodkar, Arpit on 1/13/22, 2:04 PM. Contains 1 Unresolved worklist item(s) and 1 Deployable artifact(s). Total number of checked out artifacts: 4, Project size: 80.04 KB.
 - NewTest**: Last commit by: Tirodkar, Arpit on 1/6/22, 5:45 PM. Contains 0 Unresolved worklist item(s) and 0 Deployable artifact(s). Total number of checked out artifacts: 8, Project size: 82.68 KB.
 - AlertProcessing**: No commits found for this project.

Figure 12: Generate Deployable

4. This step generates a deployable (an EAR file) for the project and hot deploys the changes to the running engine. No need to redeploy the changes.
5. Click on "Back to Home Page" option on the top right to view the endpoints of SDS.

The screenshot shows the TIBCO CLOUD Events WebStudio home page. It features the same navigation and search bar as the dashboard. A prominent yellow warning box displays the message: "Deployable generated successfully and hot deployed to running engine. [Note: If the services in this project have changed since the last deployment, a manual redeployment of the application is required for these changes to take effect!]. Below the message, there are sections for Recent User Activity and Project Snapshots, along with the counts for total unresolved item(s) (2) and total deployable artifact(s) (15).

Figure 13: Home page - view Endpoints option

The next section describes the steps for Testing the Decision Service.

Testing the Decision Service

Once the deployable is generated and deployed, the decision service can be tested using the exposed REST API Endpoints.

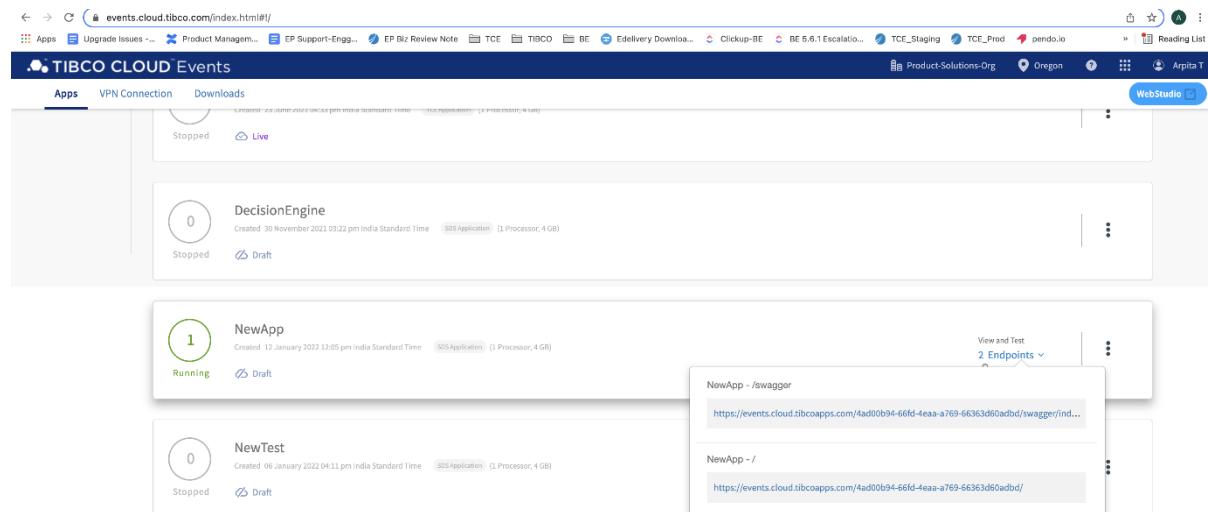


Figure 14: REST Endpoints

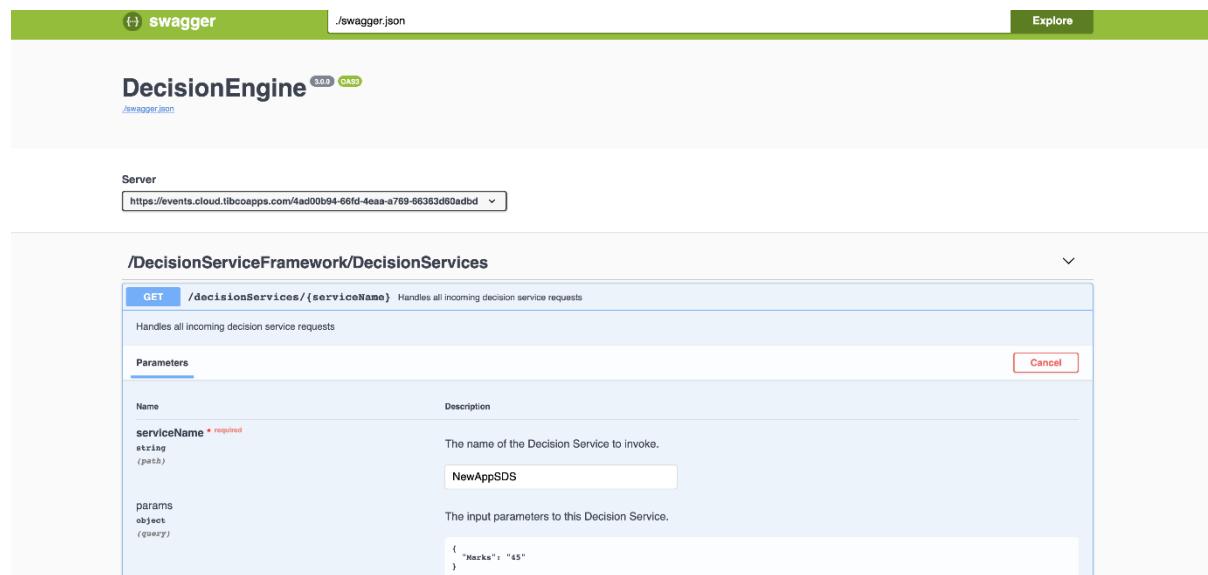


Figure 15: Swagger Client

Note:

- The serviceName is case-sensitive and expects the name as entered while creating the decision service in the WebStudio.

- The input and output are both in the JSON format.

Server response

Code	Details						
200	<p>Response body</p> <pre>{ "attributes": { "id": 6, "type": "www.tibco.com/be/ontology/DecisionServiceFramework/Events/ServiceResponse" }, "message": "The service matched 1 row(s)", "payload": { "response": { "attributes": { "type": "www.tibco.com/be/ontology/DecisionServiceFramework/Events/ServiceResponse" }, "outputs": [{ "attributes": { "id": 2, "type": "www.tibco.com/be/ontology/DecisionServices/NowAppSDS/NowAppSDSResponse" }, "Result": "Pass" }] } } }</pre> <p>Download</p> <p>Response headers</p> <pre>access-control-allow-origin: * cache-control: max-age=0, no-cache, no-store, must-revalidate content-length: 554 content-type: application/json; charset=utf-8 date: Thu, 13 Jan 2022 11:33:23 GMT expires: - pragma: no-cache strict-transport-security: max-age=31536000; includeSubDomains; preload x-content-type-options: nosniff x-frame-options: SAMEORIGIN x-xss-protection: 1; mode=block</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>200 success</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	200 success	No links
Code	Description	Links					
200	200 success	No links					

Figure 16: Swagger Output

Summary

- As you've noticed, creating a Simple Decision Service via TIBCO Cloud Events is straightforward and provides a strong rules and decisioning capability via REST APIs which can be easily integrated with any other systems or applications within an organization.
- For any feedback, queries or additional information, please reach out to <mailto:support@tibco.com>

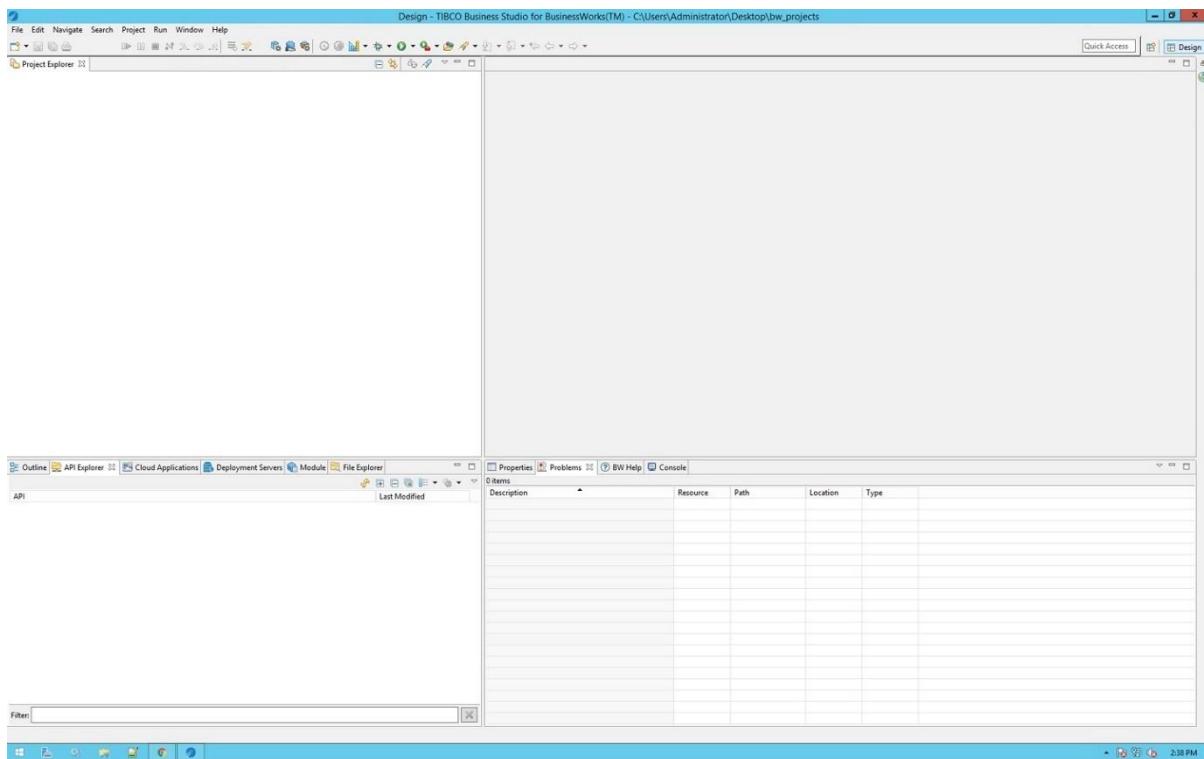
Extra Lab: BusinessWorks Apps in TIBCO Cloud Integration

In this lab, you'll first connect TIBCO Business Studio™ for BusinessWorks from your workstation to TIBCO Cloud. You'll then import a BusinessWorks project into Business Studio that implements a system API that gets purchase orders from SAP Hana. Finally, you'll push the BusinessWorks app to TIBCO Cloud and test it.

1 Getting Ready

To get ready, open TIBCO Business Studio™ for BusinessWorks.

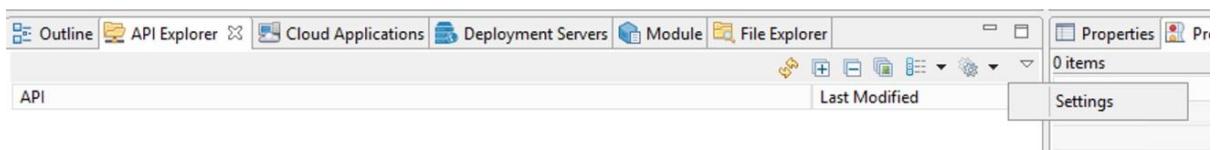
Assuming you start with a clean environment, your screen should look similar to this:



2 How to Do It: Connect Business Studio to TIBCO Cloud

To setup a connection from TIBCO Business Studio™ for BusinessWorks to TIBCO Cloud that enables you to push an app to the cloud directly from TIBCO Business Studio, do the following:

1. Go to API Explorer, click the down arrow in the upper right corner, and click Settings.



2. In the **Configure API Settings** form, click the **New** button. Fill out the **URL** and your TIBCO Cloud Integration username and password in the new form.

Field	Value
URL	<code>https://integration.cloud.tibco.com:443</code>
Username	<code>your@email.com</code>
Password	<code>YoUrPassWoRd</code>

3. Your screen should look like this:

The screenshot shows the 'Edit' configuration dialog box for API Registry client configuration. The window title is 'Edit'. The main title inside the window is 'Edit' and the subtitle is 'Edit API Registry client configuration.'

Configuration fields:

- Name: Cloud
- Type: Cloud Local Folder
- URL: https://integration.cloud.tibco.com:443

Tab selection:

- Basic (selected)
- Advanced
- Accounts

Authentication section:

- Oauth2 URL: https://sso-ext.tibco.com/as/token.oauth2
- Username: @
- Password: [REDACTED]

Buttons at the bottom:

- Finish
- Cancel

4. Click the **Finish** button when you done done, and click the **Close** button in the **Configure API Settings** form. The API Explorer should look like this:



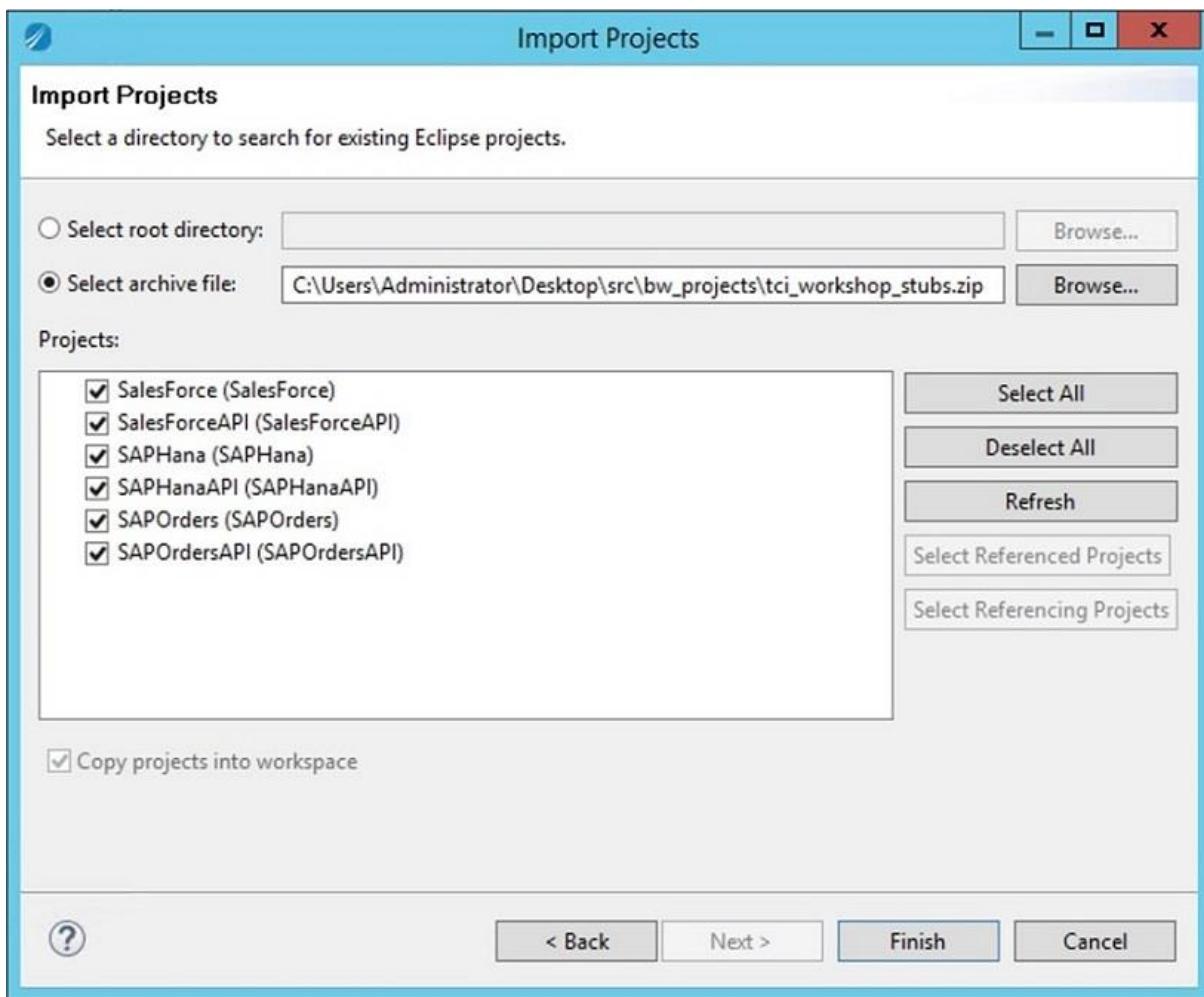
The screenshot shows the TIBCO API Explorer interface. On the left, there is a tree view labeled "API" containing several entries: SAPHanaPurchaseOrders (1.0), SAPOrders (1.0), SalesOrders (1.0), TIBCO Cloud Integration PetStore Sample (1.1), and sfContacts (1.0). On the right, there is a table with columns "Last Modified" and the corresponding dates: 05-09-19 03:31:18, 05-09-19 03:31:18, 05-09-19 01:39:05, 05-08-19 14:18:18, and 05-09-19 03:31:19 respectively.

Last Modified
05-09-19 03:31:18
05-09-19 03:31:18
05-09-19 01:39:05
05-08-19 14:18:18
05-09-19 03:31:19

3 How to Do It: Import a Project in to Business Studio

To import a BusinessWorks project into Business Studio that implements a system API, do the following:

1. From the menu, select **File > Import**. In the new form, select **General > Existing Studio Projects into Workspace**.
2. Select the [tci_workshop_stubs.zip](#) file from the [src/bw_projects](#) directory. Your screen should look like this:



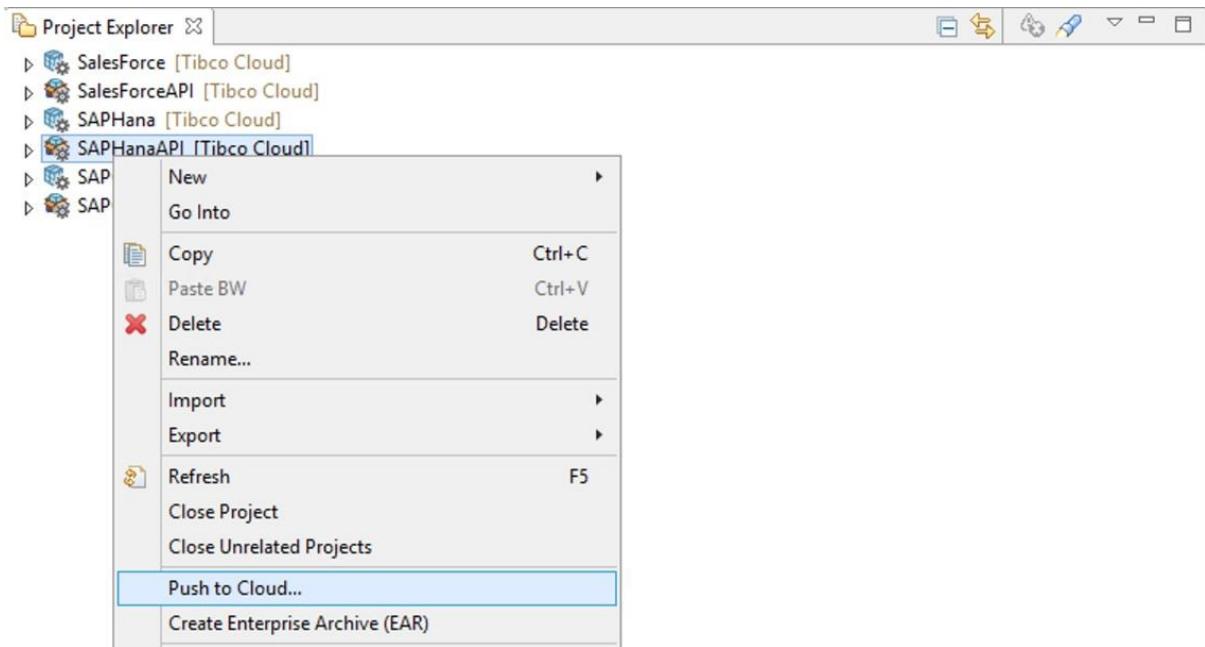
3. Click the **Finish** button. The Project Explorer should look like this:



4 How to Do It: Push the Project to TIBCO Cloud and Test

To push the SAP Hana system API app to TIBCO Cloud, do the following:

1. In the Project Explorer, select **SAPHanaAPI > Push to Cloud ...:**



2. Navigate to TIBCO Cloud, and once the app has been successfully pushed, your screen should look like this:

3. Once the **SAPHanaAPI** app is running, hover over the **Endpoint** link, and select **View and Test** from the menu
4. Test the **SAPHanaAPI** app by filling out a value in the **orderId** field, and clicking on the **Try it out!** button:

TIBCO CLOUD Integration

Apps API Specs Connections Extensions VPN Connections Downloads

SAPHanaPurchaseOrders (Tester)

See more

Search resources

Collapse all | Expand all

default

GET /purchaseOrders/{orderId}

ShowHide List Operations Expand Operations

Response Class (Status 200)
Success response
Model Example Value

```
{ "g": { "AssignmentReference": "string", "CompletedDeliveryIsDefined": true, "CreatedByUser": "string", "Customer": "string", "CustomerPaymentTerm": "string", "CustomerPurchaseOrderDate": "string", "CustomerPurchaseOrderType": "string", "DeliveryBlockReason": "string", } }
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
orderId	123		path	string

Try it out!

5. Copy the URL, which has a pattern

like <https://integration.cloud.tibcoapps.com:443/xxxxx/PurchaseOrders/purchaseOrders/{orderId}>, and store it somewhere. You will need it when you implement the process API in the next lab.

