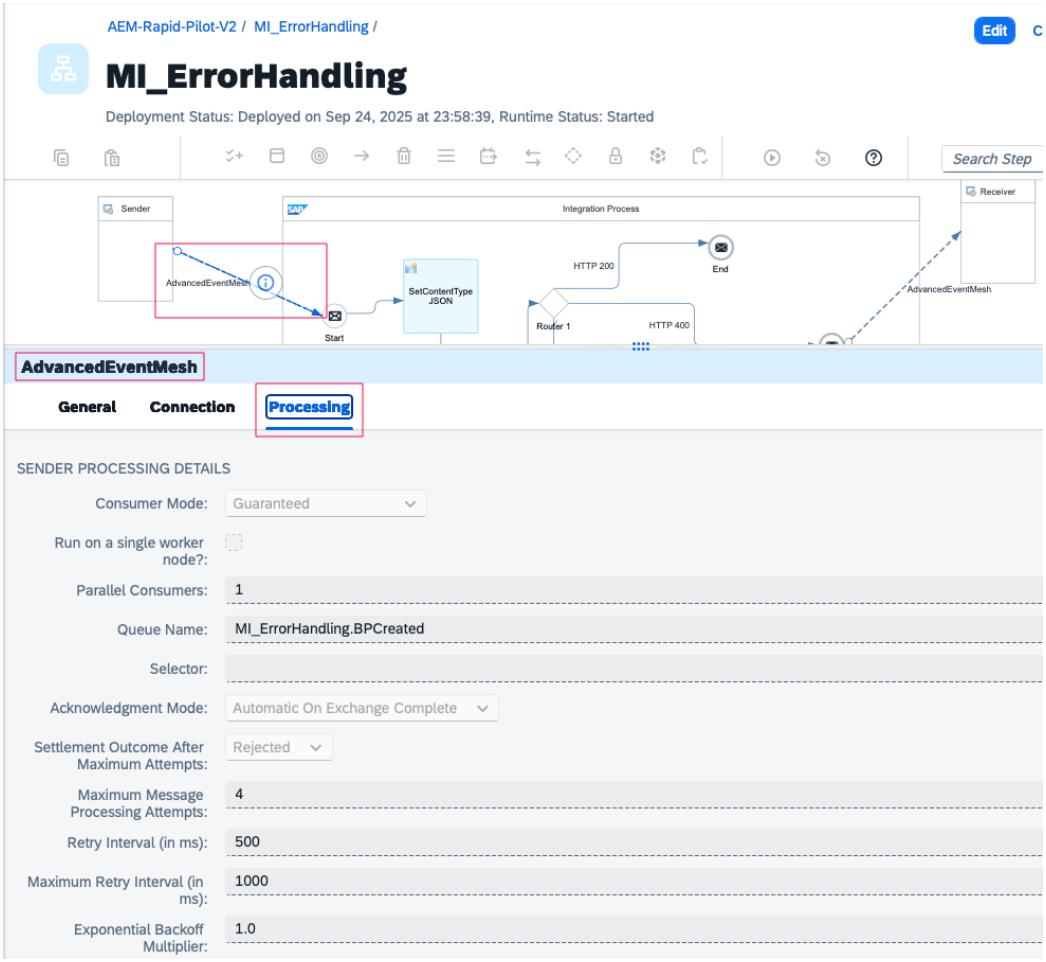
Table of Contents

1 AEM Adapter Error/Retry handling settings

There are times when a message becomes undeliverable to the intended destination. We do not want these kinds of messages to block processing, so we use the built-in error/retry-handling on the AEM adapter and/or AEM queues to achieve this without having to write any code!



Let's look at these settings one by one:

1. **Acknowledgement Mode: "Automatic on Exchange Complete"**

The most important setting when it comes to not accidentally acknowledging and therefore removing a message from the broker's queue. This setting tells the flow/AEM adapter to only acknowledge (ack) the message after the flow has successfully completed processing the message. If any error in the processing occurs, the AEM adapter will instead send a negative acknowledgment back (nack) to tell the broker to keep the message and retry it, because it couldn't be successfully processed by the flow. The alternative is to immediately ack the message when it's received, which will always result in the message being removed from the queue even if the flow fails to successfully process the message. (!!)

2. **Settlement Outcome After Maximum Attempts: "Failed"**

This setting controls the nack type and behaviour, we have two options here:
a) **Failed**, which will nack the message back to the broker and lets the broker check the retry count of the message to trigger retries based on the queue settings and only sending messages to DMQ when the retry count on the message has exceeded the max retry settings on the queue.
N.B. With the Failed setting, values of the Maximum Redelivery Count on the queue **and** the max. message processing attempts on the adapter are taken into account.

b) **Rejected**, which will nack the message telling the broker to immediately move the message to DMQ when the AEM adapter settings (Maximum Message Processing Attempts) are exceeded irrespective of queue settings.
N.B. With the Rejected setting, only the value for max. message processing attempts on the adapter is taken into account.

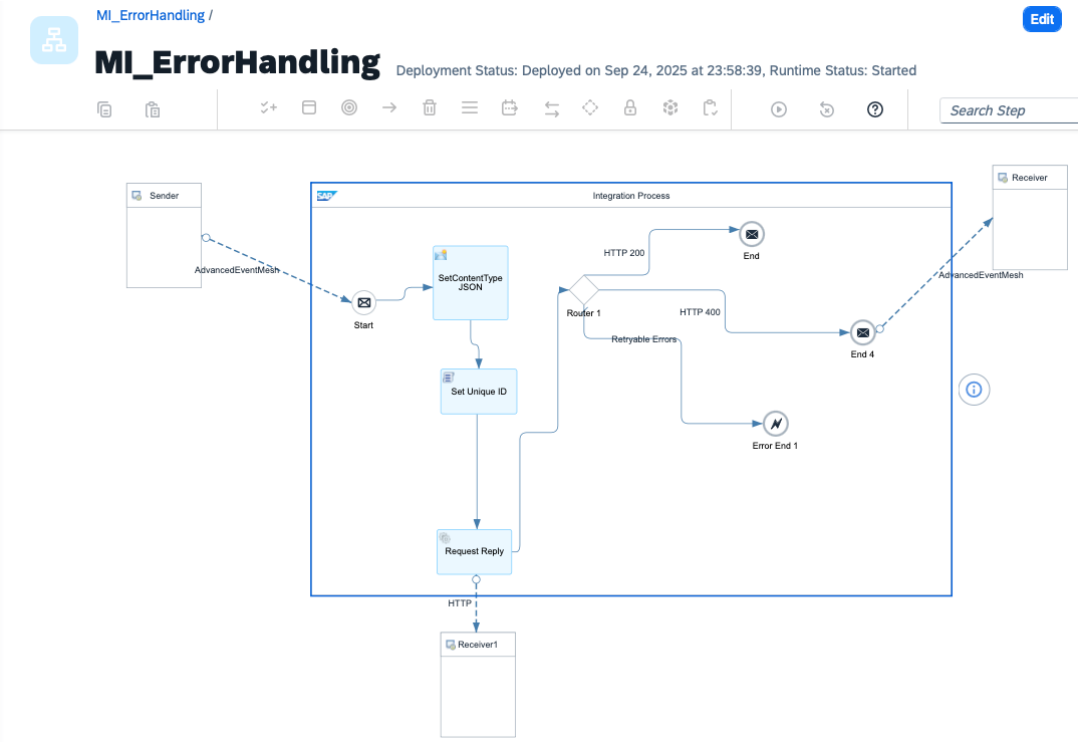
3. **Max. Message Processing Attempts: 2**

Controls how often we want to retry a message inside the iFlow before we "give up" and pass it back to the broker.

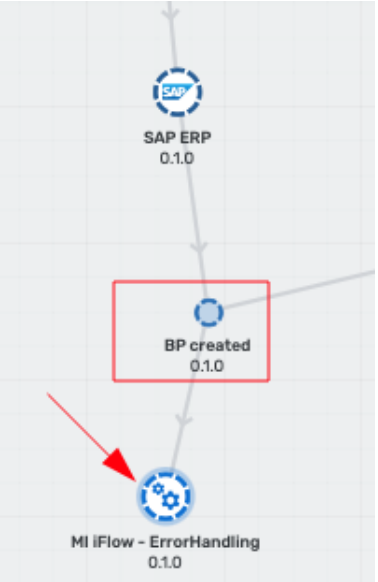
4. **Retry interval, Max Retry Interval and Exponential Backoff Multiplier**

These are all settings that control how quickly we want to retry and whether we want to incrementally increase our retry delay with each failure. A good retry delay value prevents the iFlow from repeatedly retrying a message within a few milli-seconds and gives some time for transient error situations to clear before we retry.
Note that the error handling and retry settings go hand-in-hand with the DMQ and retry settings on the input queue for this flow (queue retry settings multiply with the internal retry settings in the iFlow, e.g. if the iFlow tries 2 times internally every time we pass it a message and the broker is configured to retry the same message 3 times to the broker, then we might get 8 executions before the message is actually stopped being processed and moved to the DMQ [(1 initial attempt + 3 times retry) * 2 times retry inside the iFlow = 8 processing attempts]):

2 Overview of MI_ErrorHandling iFlow



In this exercise, the requirement is to call a REST API endpoint and handle any unexpected error responses returned by the endpoint. The iFlow subscribes to ***BP Created*** events (Golden records) published from the SAP ERP system and sends these to the REST API endpoint.



Under normal circumstances, the API service will respond with a 200 ok status code. However, we have intentionally configured the API service in such a way that sometimes, it will be temporarily unavailable. At other times, it will classify a received message as badly formatted and cannot be processed. The other scenario baked into the service is that it will sometimes appear to be permanently down.

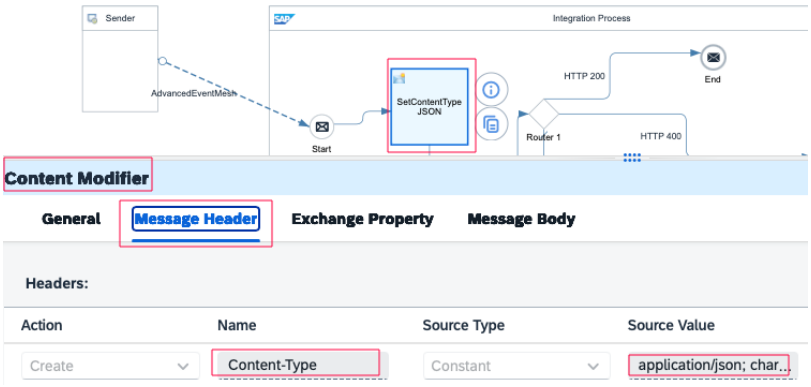
Any of these scenarios could happen in a real business situation and so the iFlow must be able to handle each one of these correctly.

The list of response codes from the API service and action required by the iFlow are shown in the table below.

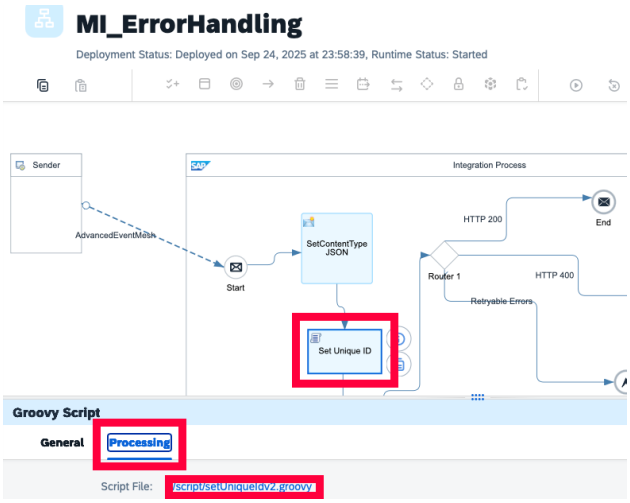
Response code from API	Description of response	Action to be taken by iFlow
200	Success	No further action required
400	Invalid message has been sent to the API	Treat as a business error and don't retry
418	Terminal error	Move to the DMQ
503	Service Temporarily unavailable	Retry for the number of times configured on the adapter/queue

The iFlow that you will download in 2.6.6 below has most of the configuration already in place (so once again, **you won't have much typing to do!**). The iFlow uses the Sender ***SAP AEM Adapter*** to get events from the `error_handling.bp_created` queue on your AEM instance.

The ***Content Modifier transformation step*** marks the content of the message as JSON so that the REST endpoint can correctly identify the message as JSON.



The ***Groovy Script transformation step*** is used to create a unique id for each deployed iFlow so that each iFlow can be identified by the mock REST service. This *unique id is only needed for the purposes of this workshop*, *and only for the scenario where the REST service is temporarily offline* and then gets reactivated.

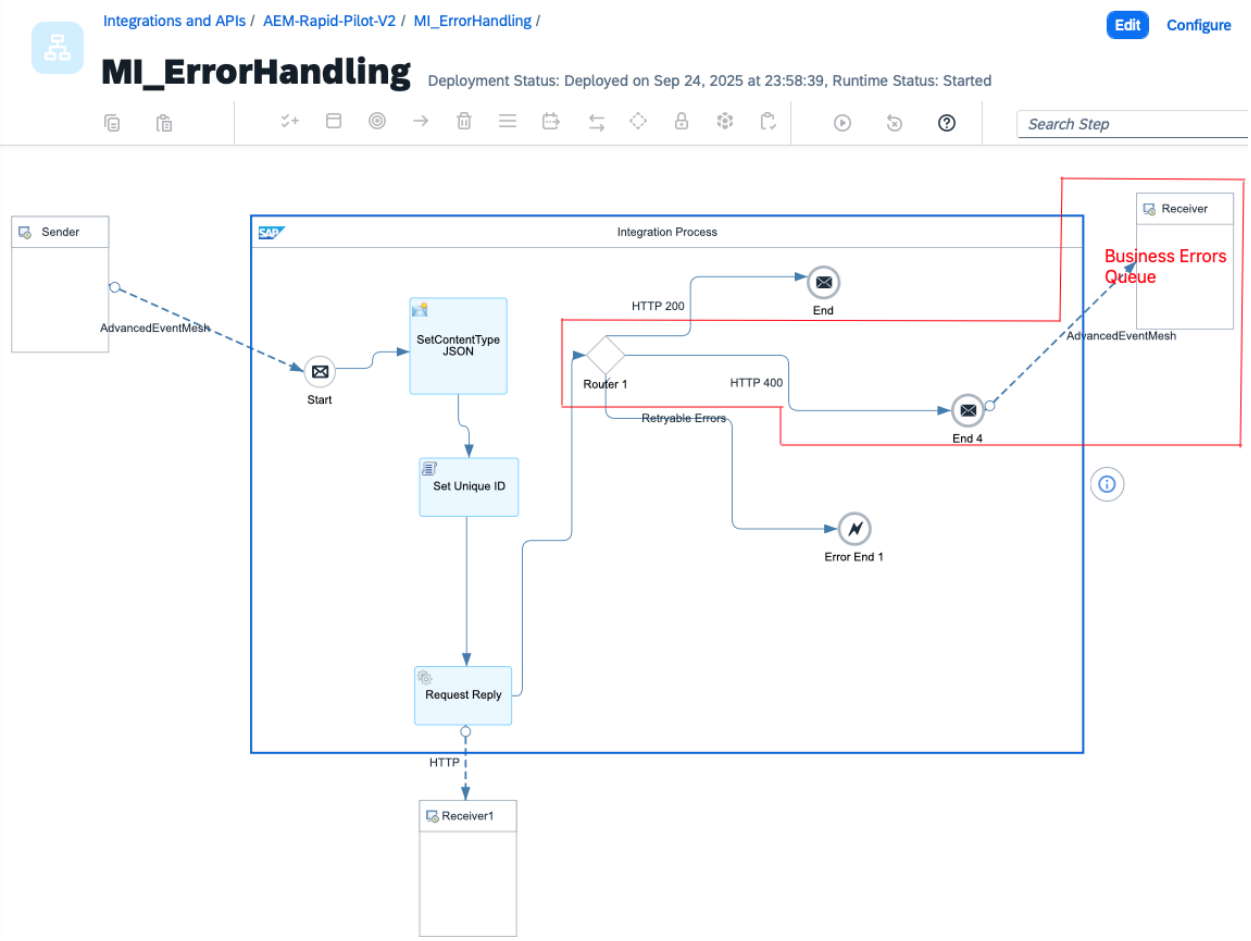


2.1 HTTP 200 response

This is the 'happy' path! No further action required by the iFlow here.

2.2 HTTP 400 response

For the workshop purposes, a 400 response is treated as a business error. A business error requires some kind of intervention by a person and so there is no point in leaving the iFlow to continually retry. In this situation, we want the iFlow to immediately remove the message from the input queue (thereby unblocking other messages) and place it on a business error queue.



To invoke the REST service so that it sends this HTTP response, the message must contain a 40000 postal code. You can either use the TryMe tab or UI5 to achieve this.

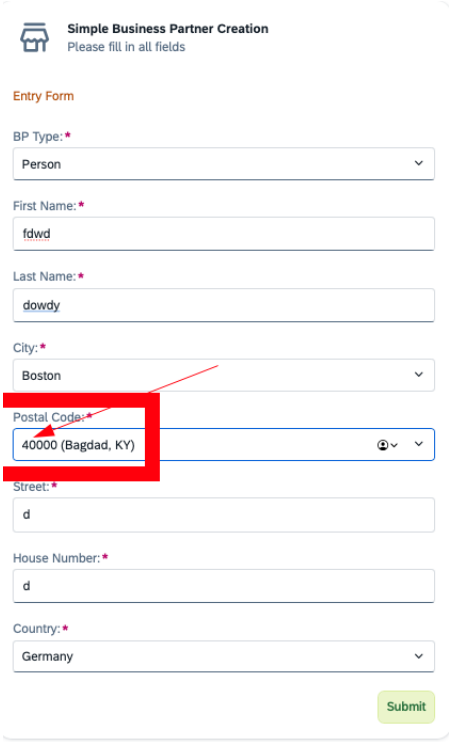
2.2.1 Invoke HTTP 400 response by using TryMe tab

You can use the sample message below in the TryMe tab to invoke a HTTP 400 response.

```
{
    "partnerId": "0001234567",
    "validTo": "2025-12-31T23:59:59Z",
    "addressNumber": "0000123456",
    "validFrom": "2025-01-01T00:00:00Z",
    "businessPartnerType": "C",
    "firstName": "Alex",
    "lastName": "Chan",
```

```
    "city": "Toronto",
    "postalCode": "40000",
    "street": "King St W",
    "houseNumber": "123",
    "country": "CAN"
}
```

2.2.2 Invoke HTTP 400 response by using the UI5 app



2.3 HTTP 418 response

For the workshop purposes, a 418 response is treated as a *terminal* error. There is nothing that can be done to rectify this error and so we want, and expect, error handling to result in the message being placed on the ***Dead Message Queue***.

To invoke the REST service so that it sends this HTTP response, the message must contain a 41800 postal code. You can either use the TryMe tab or UI5 to achieve this.

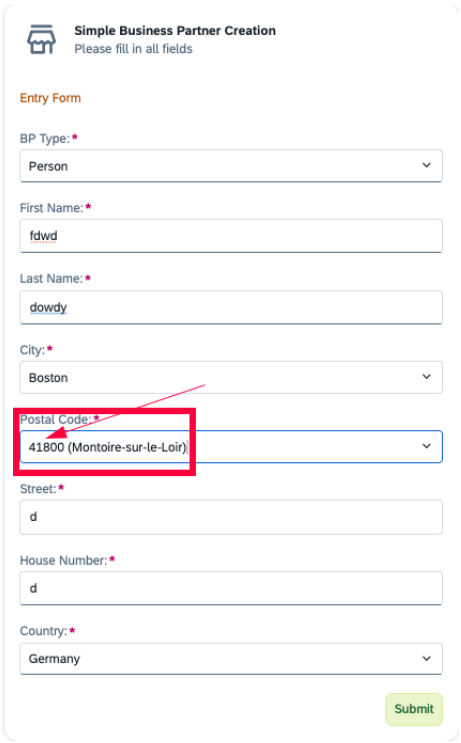
2.3.1 Invoke HTTP 418 response by using TryMe tab

N.B. **Don't forget to set the message as DMQ Eligible if you're using the TryMe Tab. The message will not get put on your DMQ if you do not set this**.

You can use the sample message below in the TryMe tab to invoke a HTTP 418 response.

```
{
    "partnerId": "0001234567",
    "validTo": "2025-12-31T23:59:59Z",
    "addressNumber": "0000123456",
    "validFrom": "2025-01-01T00:00:00Z",
    "businessPartnerType": "C",
    "firstName": "Alex",
    "lastName": "Chan",
    "city": "Toronto",
    "postalCode": "41800",
    "street": "King St W",
    "houseNumber": "123",
    "country": "CAN"
}
```

Invoke HTTP 418 response by using the UI5 app



2.4 HTTP 503 response

For this workshop, a *HTTP 503* response is treated as a *temporary* error. The service will respond with a success response after several retries. We have engineered the REST service so that it will respond with 3 *HTTP 503* responses and then a *HTTP 200* response on the 4th attempt. The message will not be placed on the ***Dead Message Queue*** even though it is retried several times.

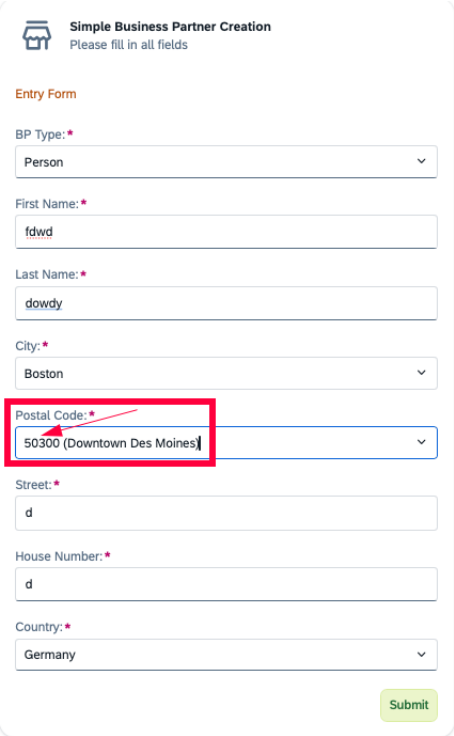
When the REST service is invoked by this iFlow and the message being sent contains a 50300 postal code, this signals to the REST service that it needs to simulate being temporarily unavailable. For each iFlow that calls the service it will respond 3 times with a HTTP 503 response. On the 4th invocation, it will respond with a HTTP 200 to indicate that the service is now back online and operating normally. This is why it is necessary for each iFlow to present a unique id when invoking the REST service so it can keep track of how many times it has been called by a specific iFlow instance.

2.4.1 Invoke HTTP 503 response by using the TryMe tab

You can use the sample message below in the TryMe tab to invoke a HTTP 503 response.

```
{
    "partnerId": "0001234567",
    "validTo": "2025-12-31T23:59:59Z",
    "addressNumber": "0000123456",
    "validFrom": "2025-01-01T00:00:00Z",
    "businessPartnerType": "C",
    "firstName": "Alex",
    "lastName": "Chan",
    "city": "Toronto",
    "postalCode": "50300",
    "street": "King St W",
    "houseNumber": "123",
    "country": "CAN"
}
```

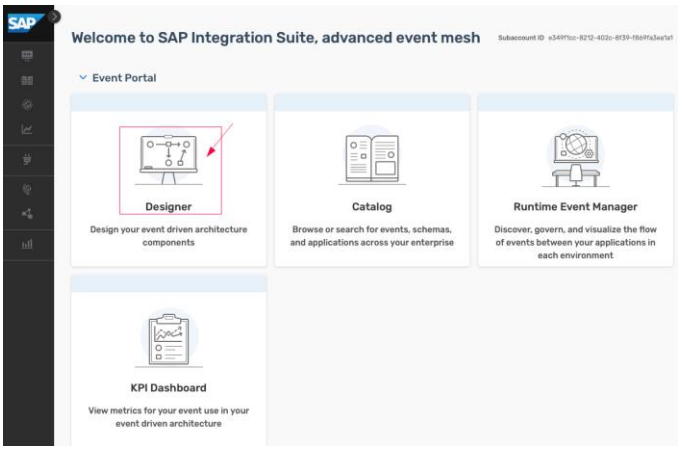
2.4.2 Invoke HTTP 503 response by using the UI5 app

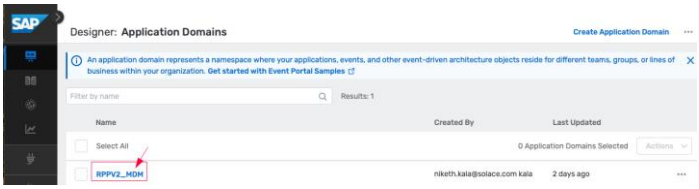


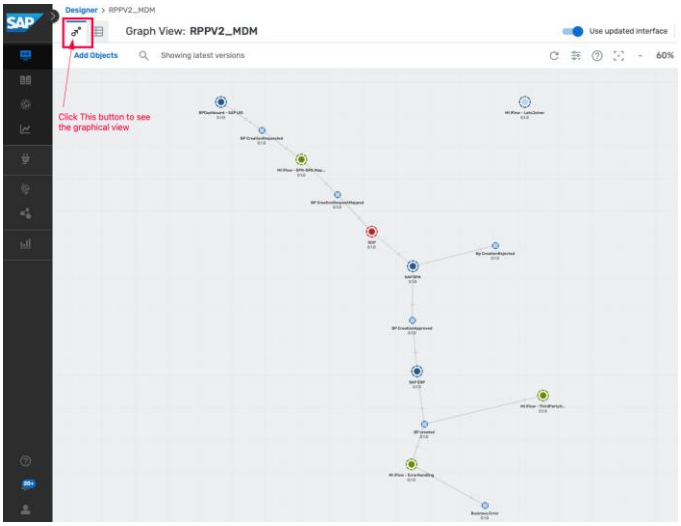
2.5 Promote supporting components from Event Portal

In the next series of steps, you will go through the process of pushing (promoting) all the required components that are required to support the deployment of the iFlow to your AEM instance. You will not be able to successfully deploy the iFlow unless all these are present. The components that will be pushed are ***A***ccess***C***ontrol***L***ists, Topics, Queues, Usernames, ACL Profiles.

2.5.1 Navigate to the Event Portal and select the Application Domain





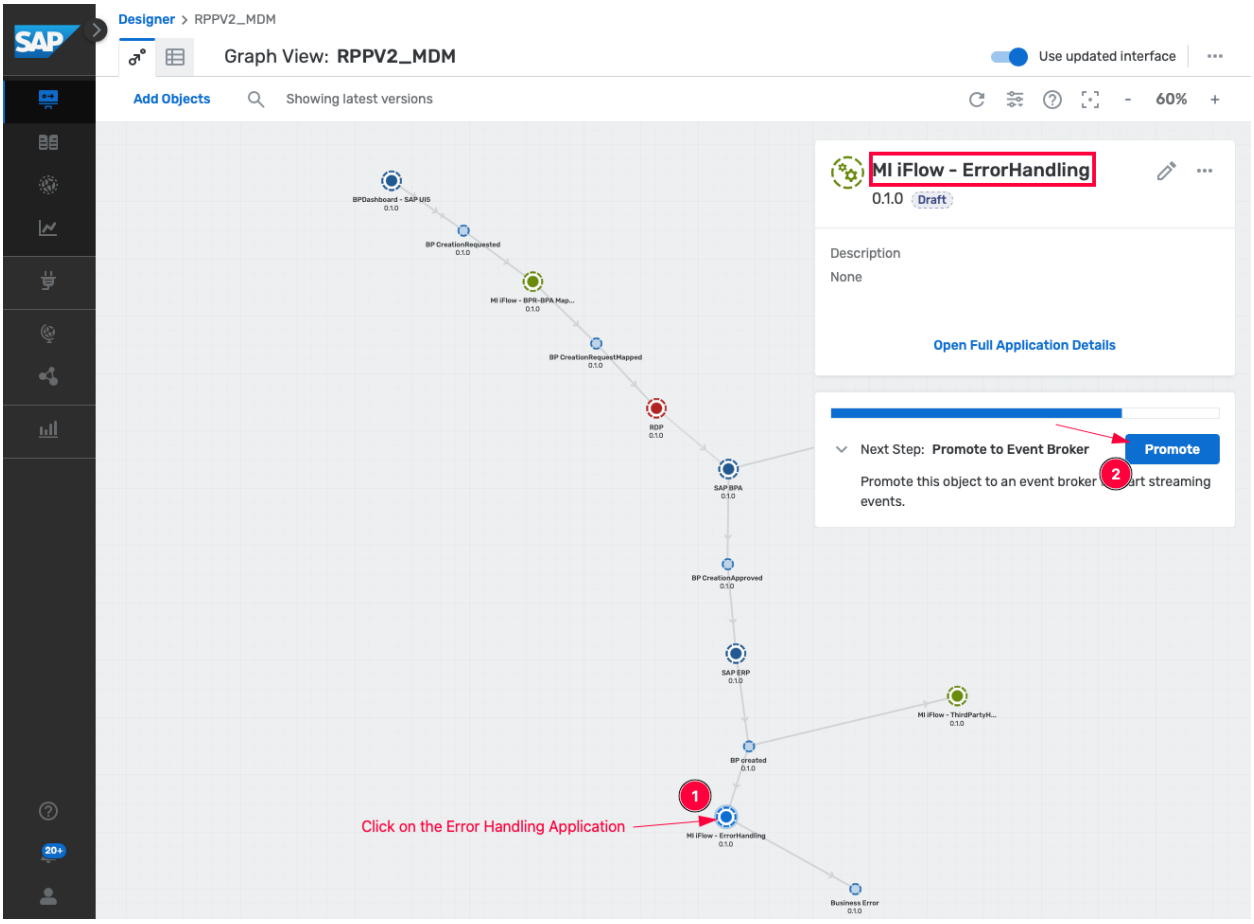


2.5.2 Application Queue Names and Client Profile Details

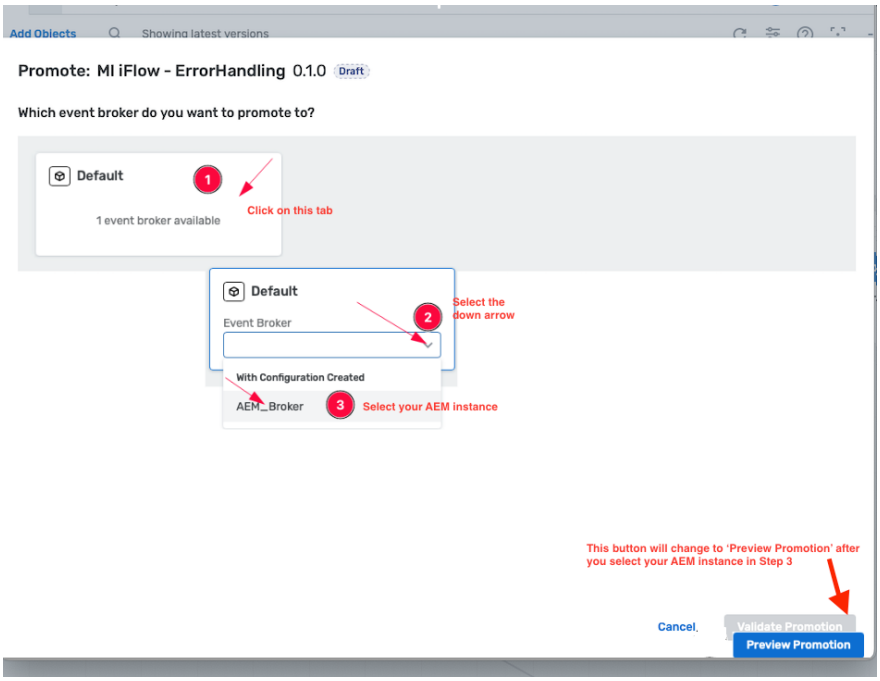
Each application's queue name and client credentials (username and password) are detailed in the table below. The configuration must follow the order indicated in the diagram. So, you must configure "*BPCreate-SAP UI5"* before "*MI iFlow BPA-BPA Mapping"* and so on…In this step, we will configure the topics and queues for the Error Handling iFlow

	Application	QueueName	ClientUsername	Pwd
1	BPCreate-SAP UI5		user1	user123$
2	MI iFlow BPA-BPA Mapping	bpr_bpa_mapping.bp_creation_request	user2	user123$
3	RDP	rdp.bp_mapped_creation_request	user3	user123$
4	SAP BPA		user4	user123$
5	SAP ERP	erp.bp_creation_approved	user5	user123$
6	3rdParty HttpEndpoint	http_endpoint.bp_created	user6	user123$
7	Error Handling	error_handling.bp_created error_handling.bp_created_dmq error_handling.bp_created_business_error	user7	user123$
8	LateJoiner	late_joiner.bp_created	user8	user123$

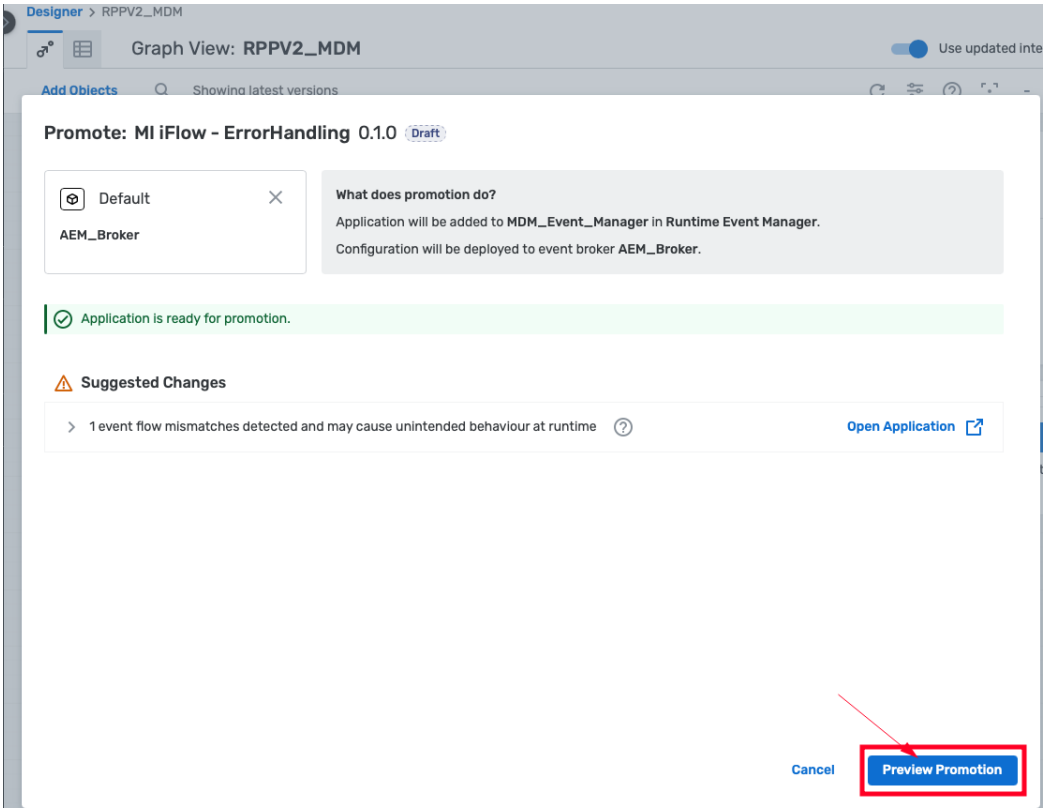
2.5.3 Open the MI iFlow Error Handling Application



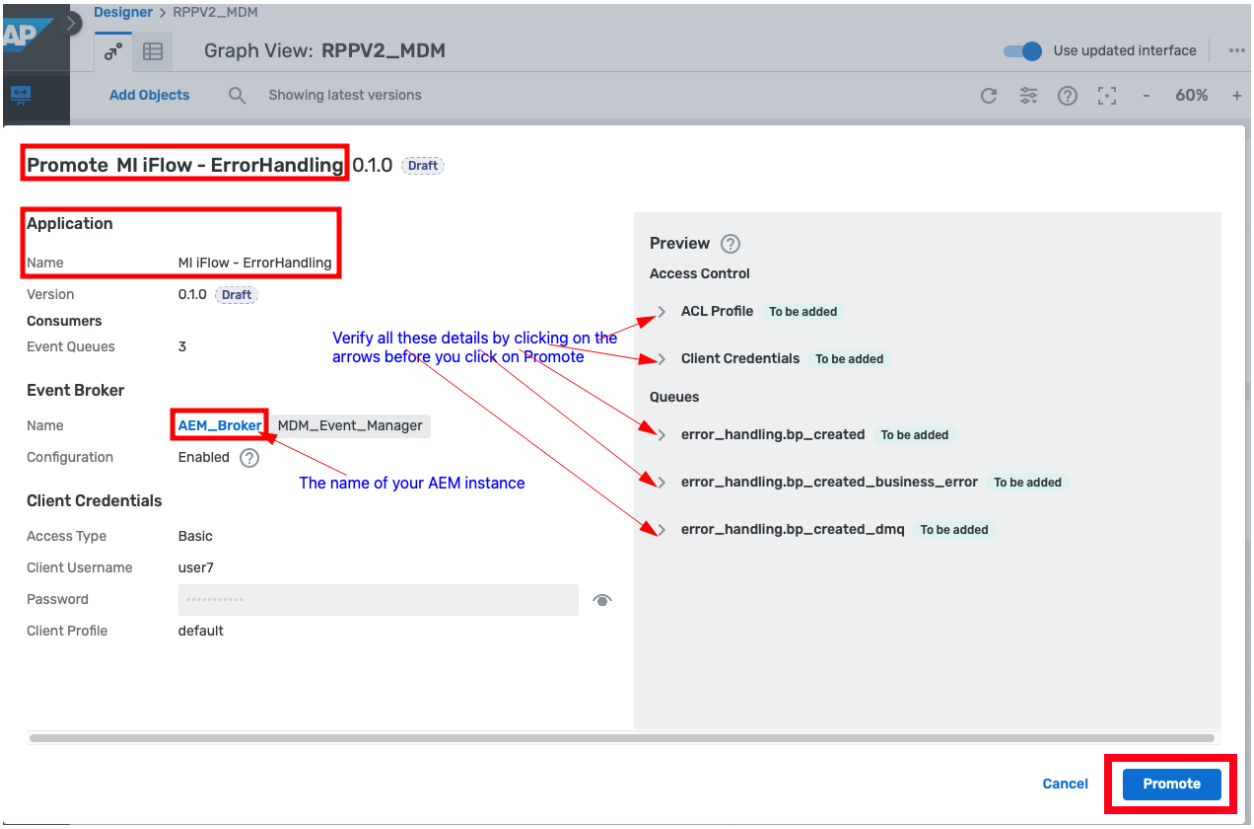
2.5.4 Select the Event Broker



2.5.5 Preview the Promotion and Validate the Application Queue Names and Client Profile Details

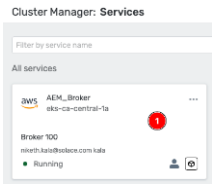


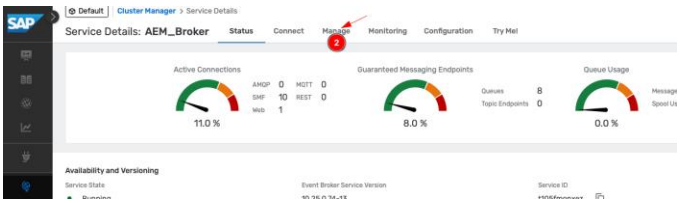
2.5.6 Validate Access Control, profile, Client Credentials, Application Queue Names before promoting

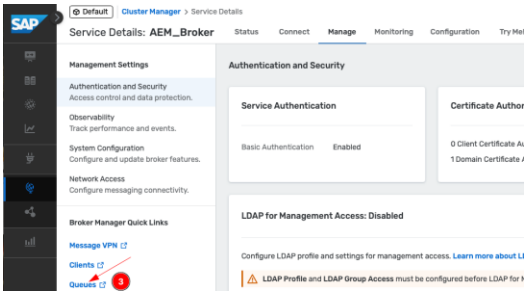


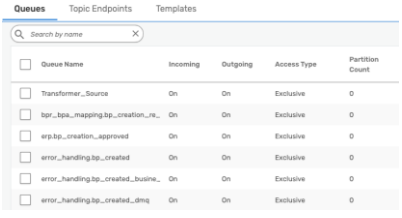
Congratulations, you have successfully promoted Application – MI iFlow ErrorHandling to your AEM Broker!

You can verify that these have all been deployed as expected by checking on your AEM instance Cluster Manager => Manage => Queues



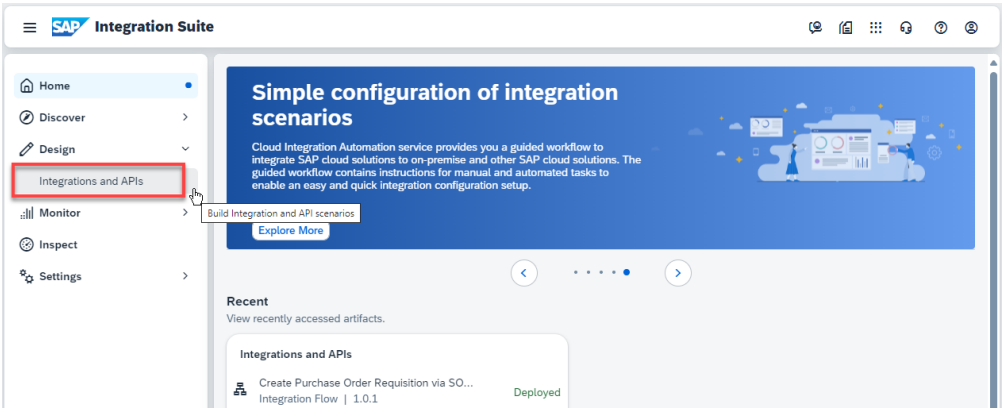






2.6 Set up packages within Integration Suite, cloud integration

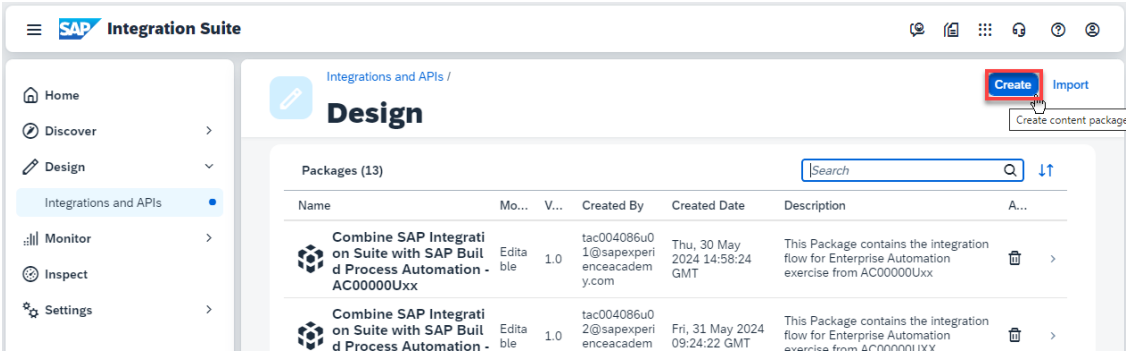
2.6.1 From your SAP Integration Suite tenant service, click on Design section in the left-hand menu. Now click on “Integration and APIs” under Design -



2.6.2 Create the integration package

After clicking on “Integration and APIs” you will see a “Design” window on the right-hand side of the screen. This is where you will create your own package.

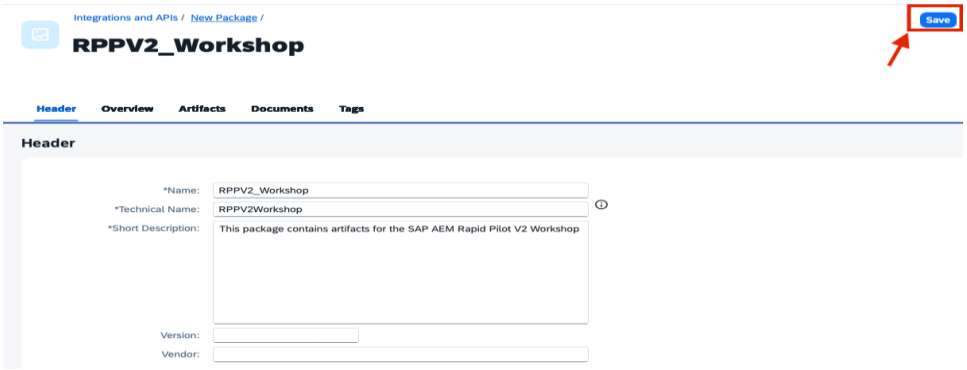
2.6.3 Click on Create to create an integration package.



2.6.4 Provide the following details:

⇒ Name: RPPV2_Workshop
⇒ Technical Name will be auto-filled with RPPV2Workshop
⇒ Short description: “*This package contains artifacts for the SAP AEM Rapid Pilot V2 Workshop.*”

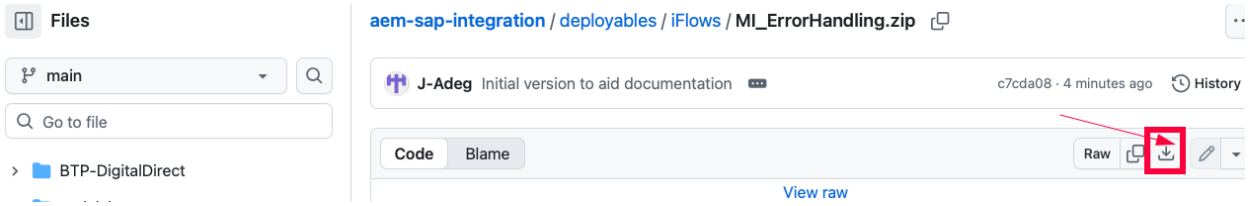
2.6.5 Click on Save once finished.



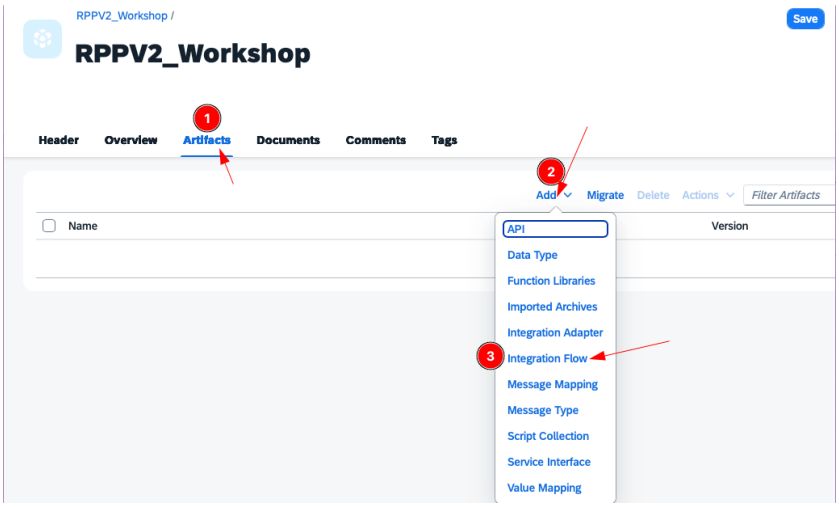
Within an integration package, you can add several artifacts: REST APIs, OData APIs, integration flows, mappings, and more. In our case, we will import templates for integration flows that will be used in the workshop.

2.6.6 Download the iFlow template for the 2nd iFlow **MI_ErrorHandling** by clicking on the download link below and then click on the Download raw file icon as shown in the screenshot below; this will download the file to the location you have configured for downloads on your browser

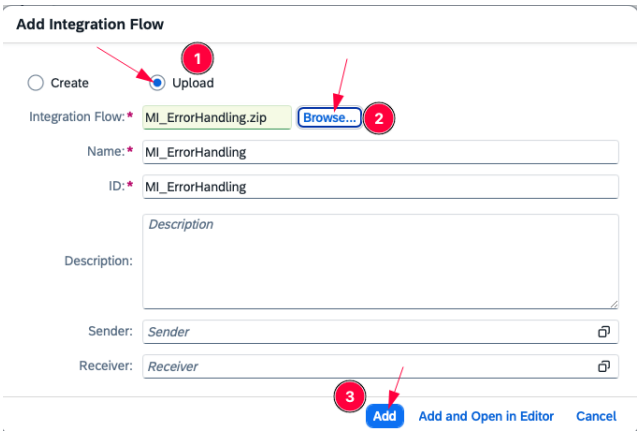
https://github.com/SolaceLabs/aem-sap-integration/blob/5191142993237af45d7140acc5208df66217ccc6/deployables/iFlows/MI_ErrorHandling_.zip



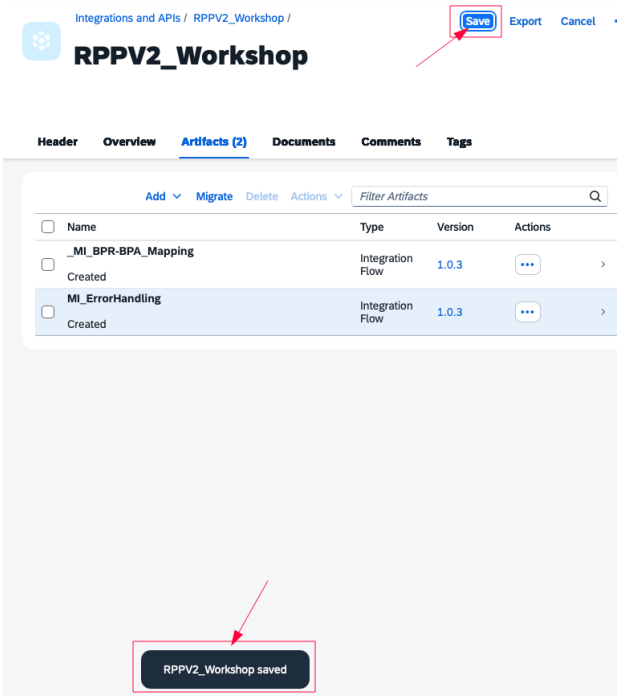
2.6.7 You now need to bring up the dialogue to import that iFlow template file that you downloaded into your package:



2.6.8 The dialogue for importing the previously downloaded file is now visible as shown in the screenshot below. Upload the file…



2.6.9 The imported iFlow template is now in your package. Save it

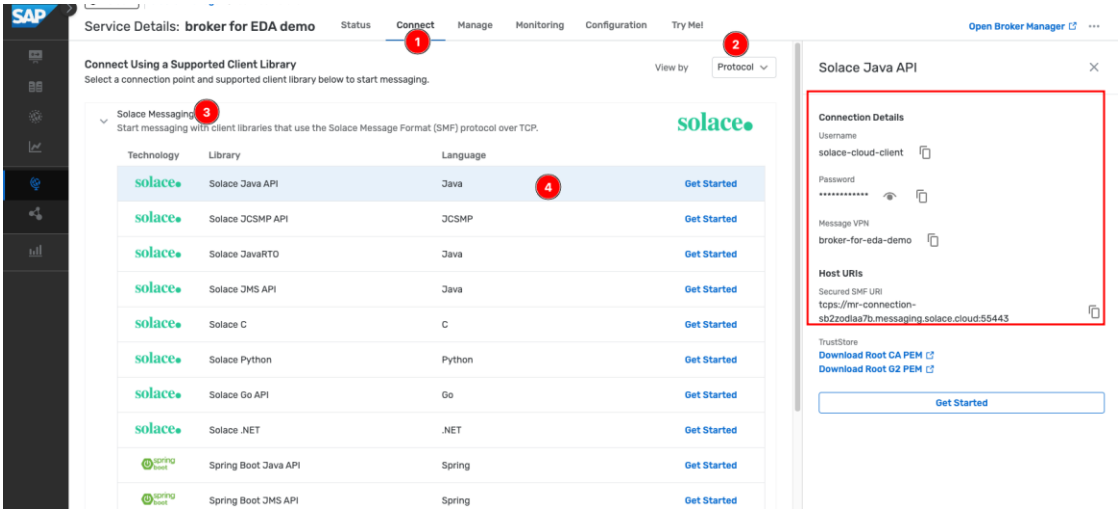


2.7 Gathering AEM instance connection credentials

The iFlow will need to be configured with the connection credentials (URL, vpn name, password) for **your** AEM instance before it can be deployed so we will gather all these details first.

2.7.1 Obtain AEM Broker Connection Credentials

Before heading back to Integration Suite, let's head to our **Advanced Event Mesh Console** and go to **Cluster Manager** -> **{your service}**. Select the connection point and protocol that you want to use to connect your Integration Suite flows by going to the "Connect" tab, order by protocol, then click on Solace Messaging. Make a note of the connectivity details underneath "Solace Messaging" (click on the section to open it up). You will need these details in the next steps when configuring your iFlows.

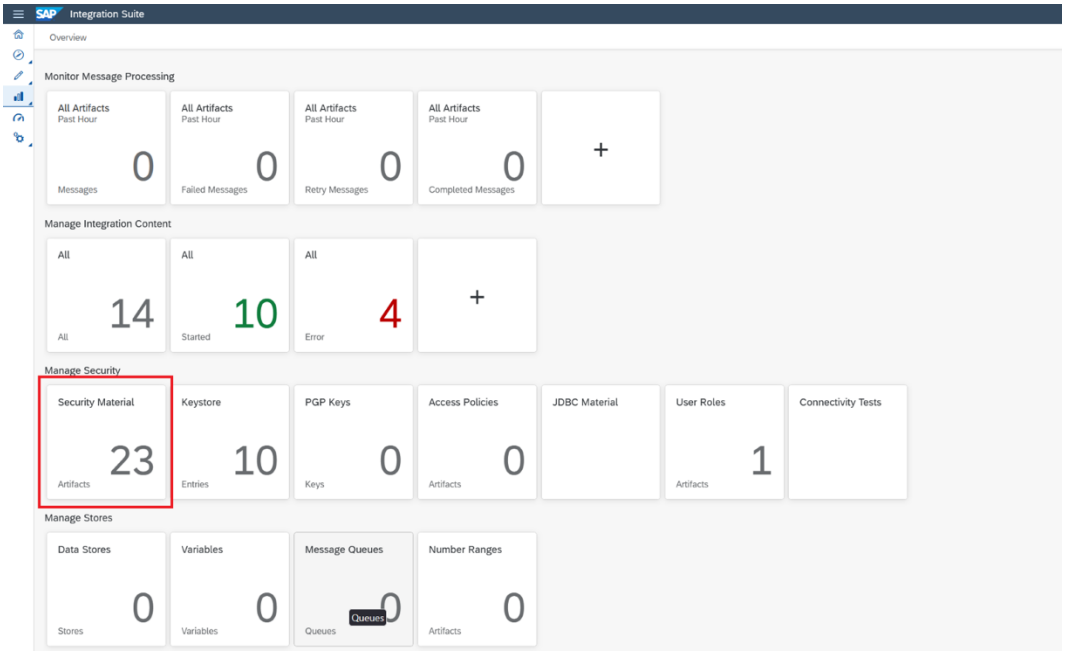


The connect tab lists all the various connectivity details for the various supported protocols. The SAP Integration AEM adapter uses Solace Messaging Format, which is AEM's very own protocol with a broad feature set support. Solace Message Format (SMF) is the underlying messaging protocol for SAP Integration Suite, advanced event mesh.

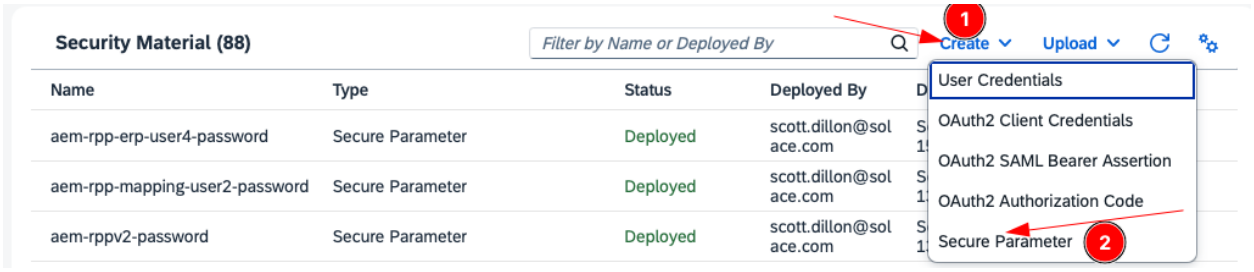
2.7.2 Security Configuration

Rather than entering the AEM instance password directly on the iFlow and making it visible to everyone with access to the iFlow, you will create a **SecureParameter** which will store the password securely and we then just reference this in our iFlows.

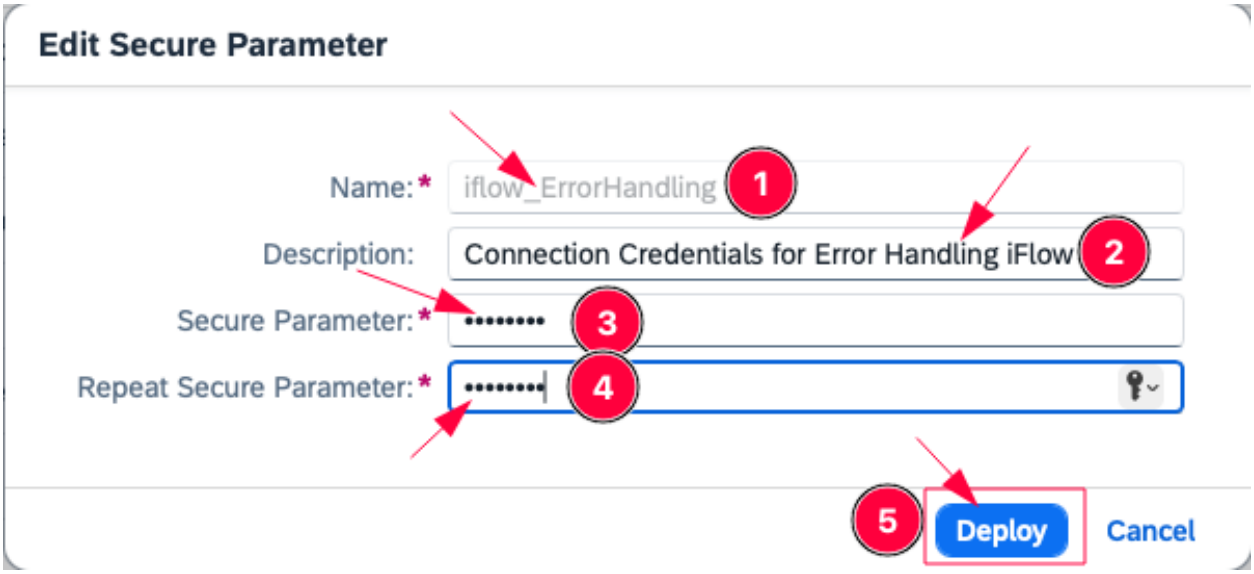
Go to Integration Suite **-> Monitor** -> **Manage Security** -> **Security Material**.



In here, create security credentials for your AEM broker service.
Create **SecureParameter** `iflow_ErrorHandling` for the iFlow:



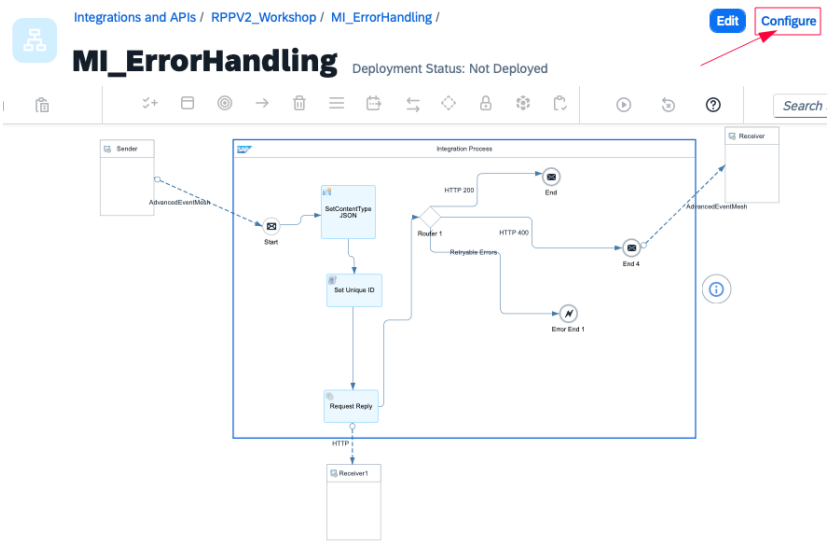
The credentials are:
Name: **iflow_ErrorHandling**
SecureParameter: **user123$**



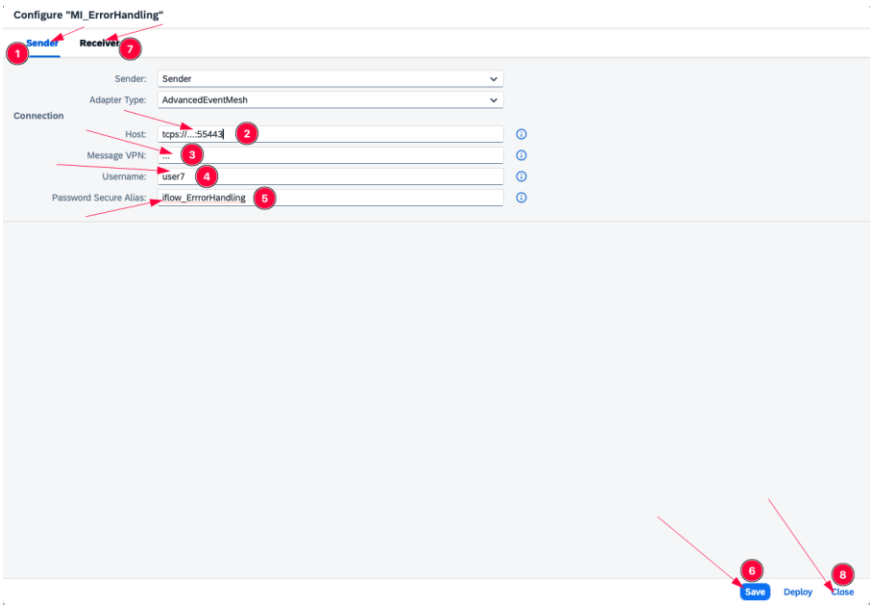
2.8 Add security and connection credentials for MI_ErrorHandling

2.8.1 Navigate to the iFlow – Go to Integration Suite -> Design -> Integrations and APIs -> RPPV2_Workshop -> Artifacts -> MI_ErrorHandling

2.8.2 Click on the Configure button to configure the AEM Adapter connection credentials

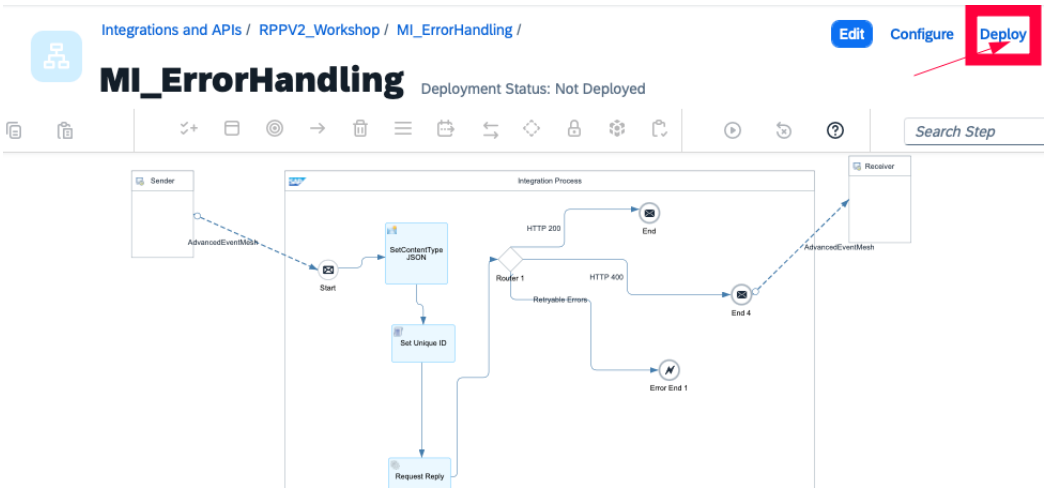


2.8.2.1 **Click on the Sender tab** and populate the connection credentials.



1. Click on the Sender tab
2. Type in the ***AEM connection url*** you obtained from 2.7.1 above
3. Type in the ***message vpn*** you obtained from 2.7.1 above
4. Type in the ***user id*** that you used to create the Secure Alias in 2.7.2 above
5. Type in the name of the Password Parameter you created in 2.7.2 above
6. Click on Save
7. **Click on the Receiver tab** and just verify that the details there are exactly as they are on the Sender tab.
8. Click on Close

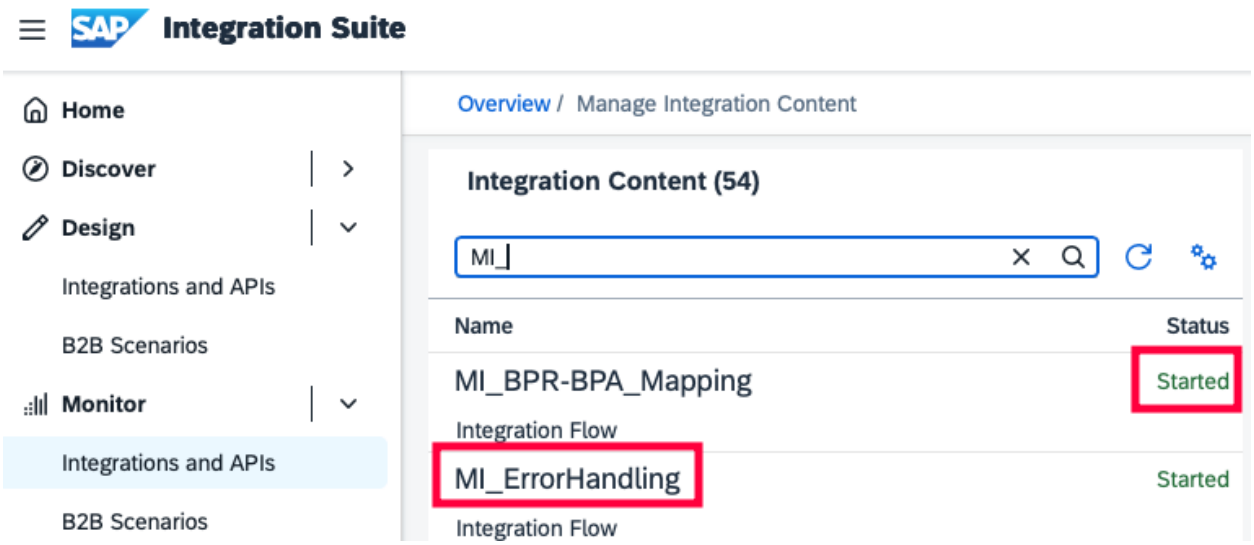
2.9 Deploy the iFlow



2.9.1 Check on the Deployment progress

Make sure the iFlow goes to a Started state. Use the instructions and screenshots below to guide you.

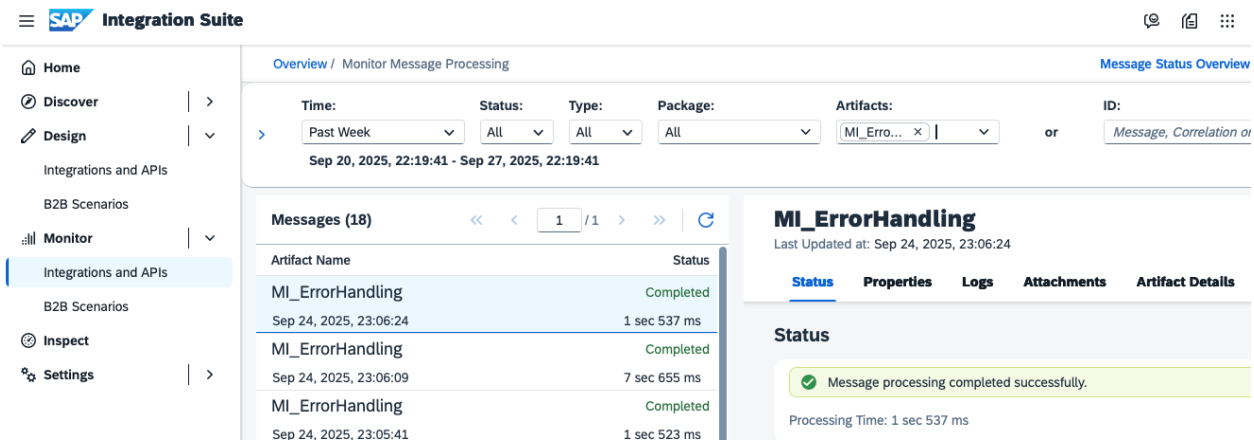
Go to **Integration Suite -> Monitor -> Integrations and APIs -> Manage Integration Content**



2.10 Test the MI_ErrorHandling iFlow

Go to **Integration Suite -> Monitor -> Integrations and APIs -> Monitor Message Processing**

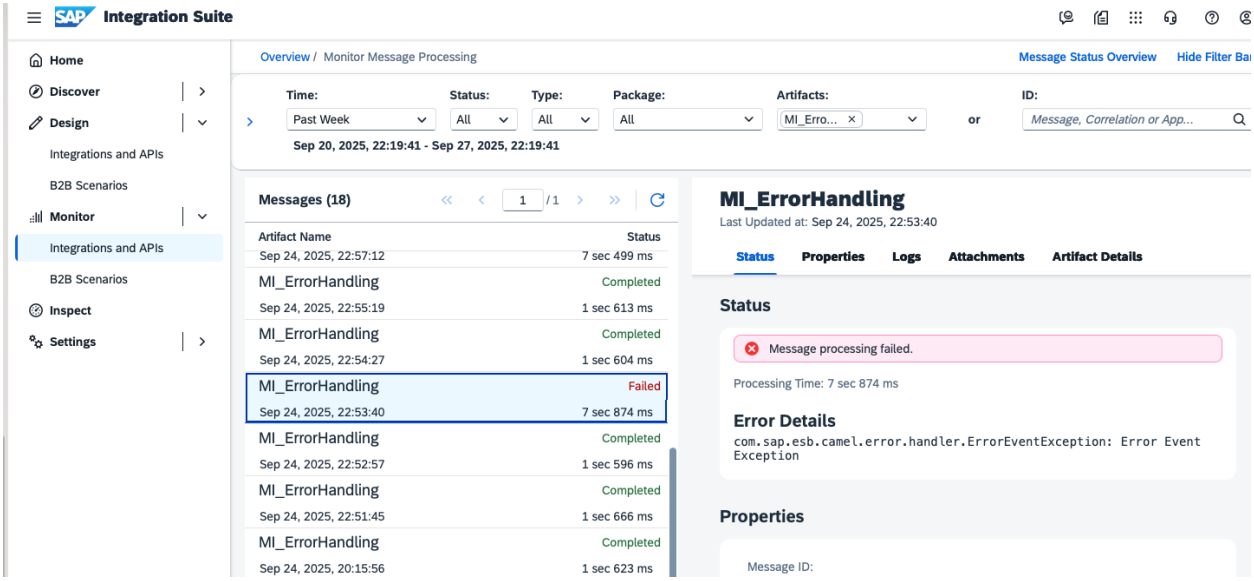
For a successful response, you should see something like:



If you don't get a response, check on the error logs for the iFlow in Integration Suite
Go to **Integration Suite -> Monitor -> Integrations and APIs -> Monitor Message Processing**

Have a look at the logs and troubleshoot any errors.

N.B. For this iFlow, you must get some failures. This is because you will intentionally be causing exceptions.

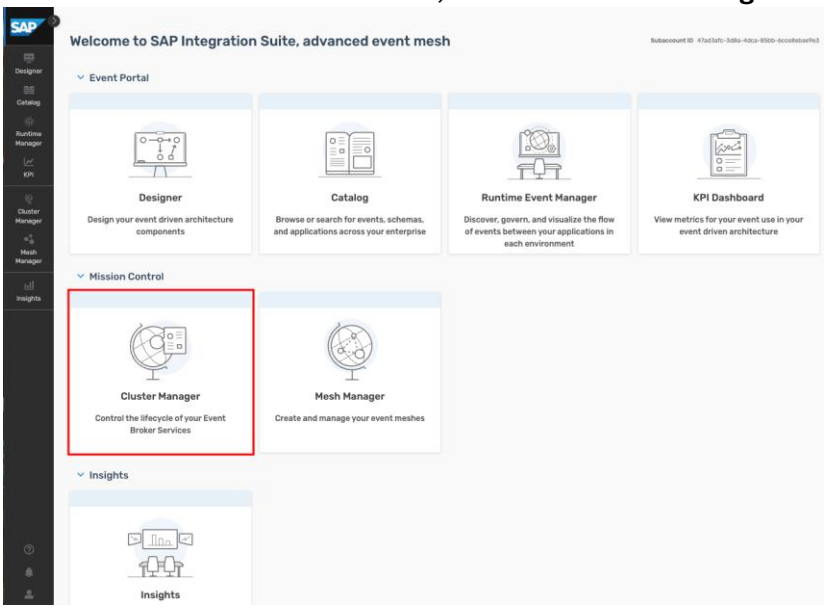


3 APPENDIX – CONNECTING TO THE TRY ME UTILITY

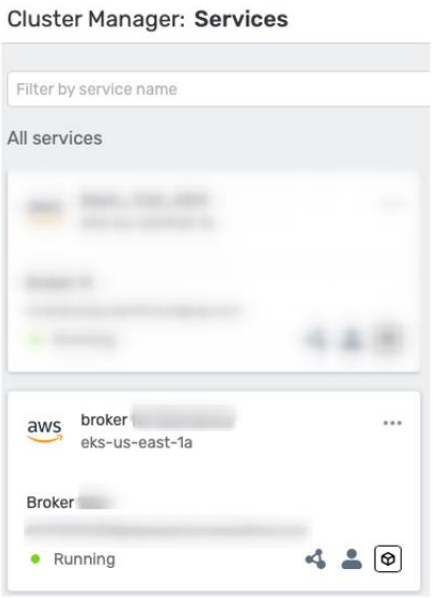
3.1 Obtain connection credentials

The Try Me! utility is a JavaScript application for quickly getting up and running with testing smart topics. The application uses Web Sockets and so you'll need to select the appropriate connection credentials.

From within the main AEM console, click on **Cluster Manager**

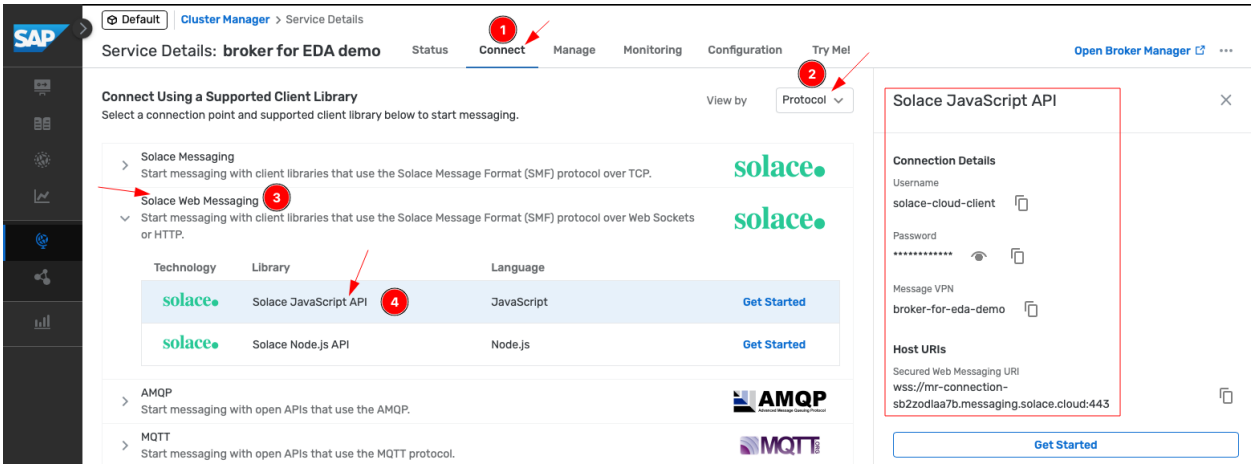


Next, select the AEM broker that you created for this workshop. The screenshot below is just an example. Your AEM instance name and Cloud Provider name will depend on what you chose!

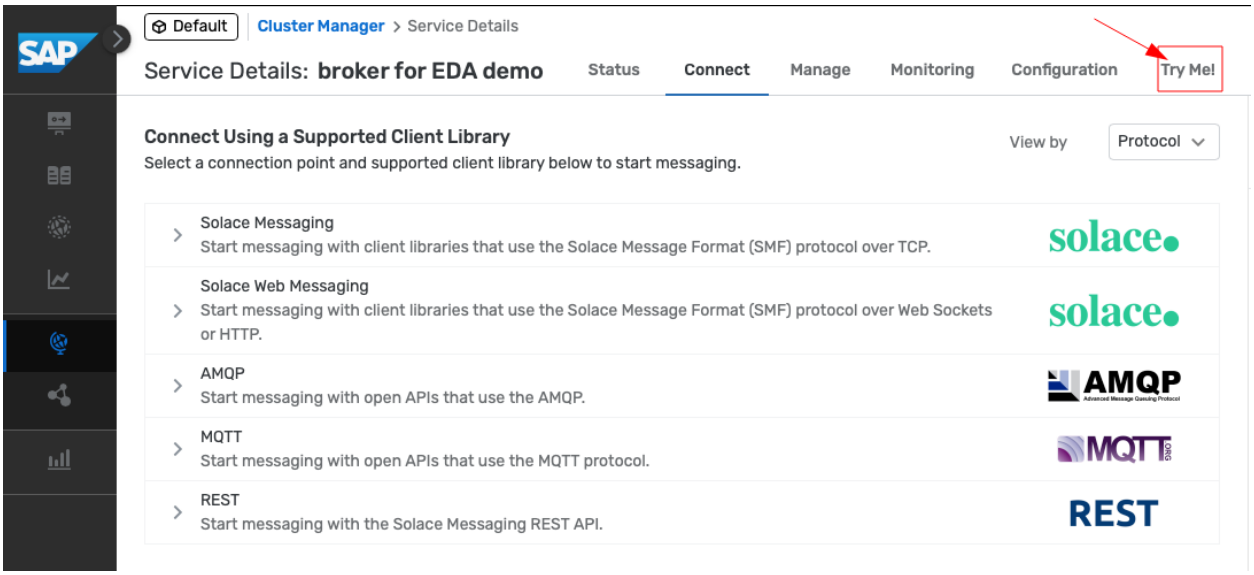


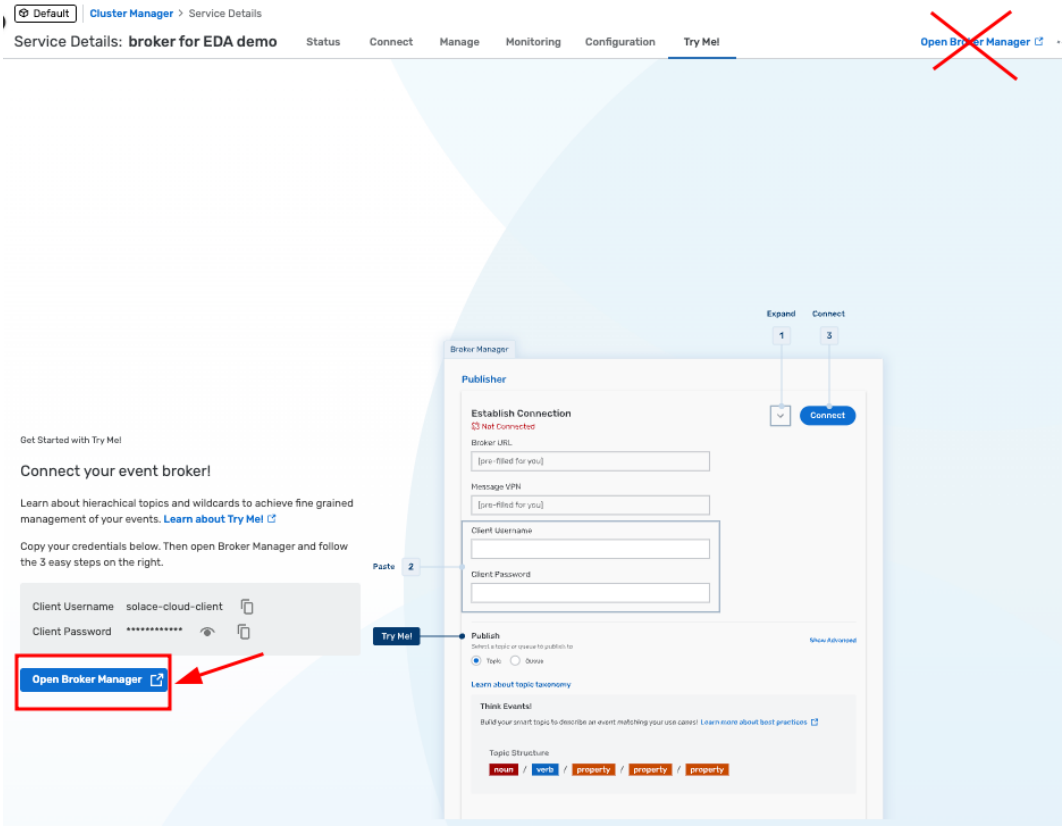
After selecting your AEM instance,
1 - click on the "Connect" tab
2 - order by protocol
3 - click on Solace Web Messaging.
4 - click on “Solace JavaScript API” to reveal the connection credentials

Make a note of the connectivity details underneath "Solace JavaScript API.

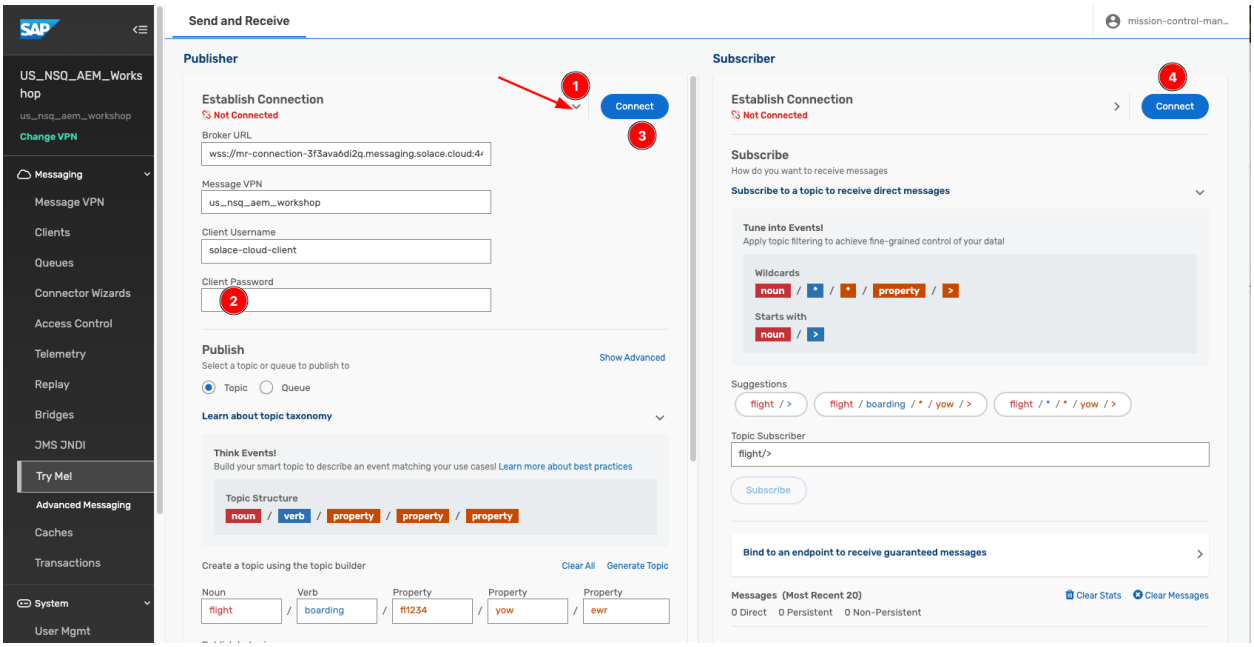


You will use these credentials within the ***Try Me Tab***. Select the ***Try Me Tab*** and proceed with “Open Broker Manager”.





Once the Broker Manager is open, select the “Try Me” option from the left side of the menu. You will then use the credentials that you copied above to populate the left side of the screen...AKA the Publisher Side. Once the publisher side says “connected”, you can simply hit the “Connect” button on the right side to also connect your subscriber.



You are now connected to the AEM service with a publisher and subscriber utilities that can be used to send/receive messages.