ABBYY Vantage

Integration with ResistantAI Fraud Detection Platform

Developed by: Nick Carr

Contents

About ABBYY Vantage Integration to ResistantAI

This integration guide describes how to integrate ABBYY Vantage with ResistantAI's fraud detection platform. This document is only to be used as a guide on how to achieve integration between both platforms. Copies of document images will be sent outside of Vantage to the ResistantAI platform to perform the fraud analysis check.

The integration has been developed using a custom activity within the workflow of Vantage, this connectivity has been developed as a demonstration of how integration with the ResistantAI platform can be achieved. The custom activity is located within a Process Skill, after the point of extraction and before a Manual Review stage, this ensures the fraud check is done before a user has seen the document.

System Requirements

You will require an ABBYY Vantage account, a valid subscription for ABBYY Vantage, and a Vantage user that is assigned the Skill Designer role to configure and to run your workflow.

You will also require a valid account for the ResistantAI platform with which you can then generate a Client ID and Secret with which to connect to their API.

Prerequisites

Environment Variables

For the process to run successfully 5 environment variables are required, these are configured within the Vantage administration interface. From the side panel in the main tenant administration interface go to:

1. Configuration
2. Environment Variables:

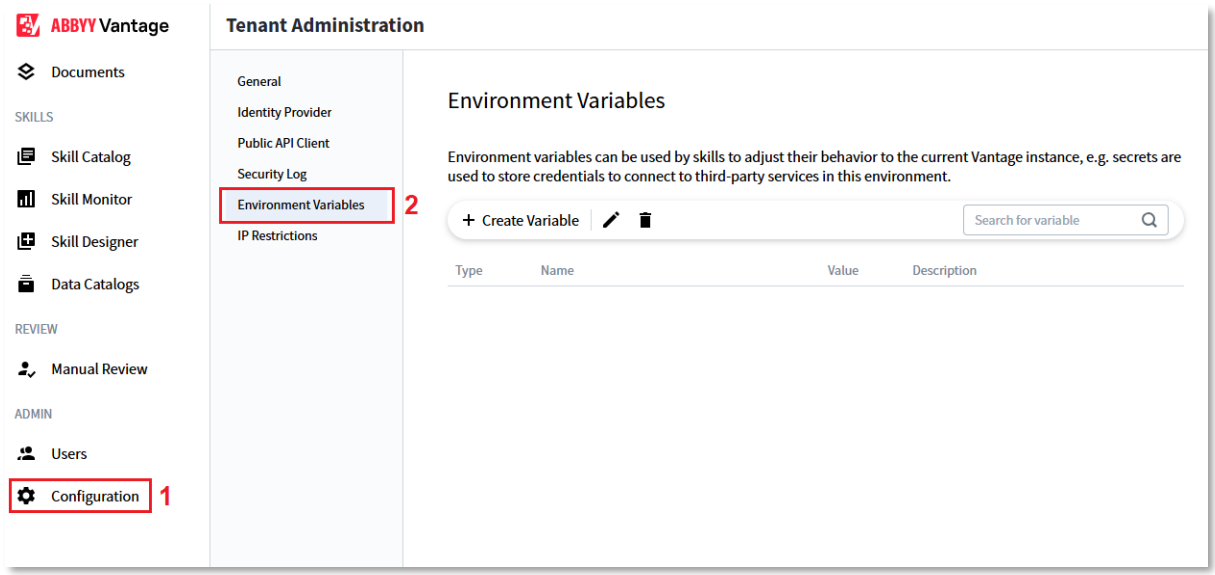

Figure 1: Tenant Administration Interface – Environment Variables.

If you do not see the option of Configuration from the left-hand panel, please contact your tenant administrator to assign the correct permissions to your login on the tenant.

To create an environment variable click on **+ Create Variable**, the following window will load:

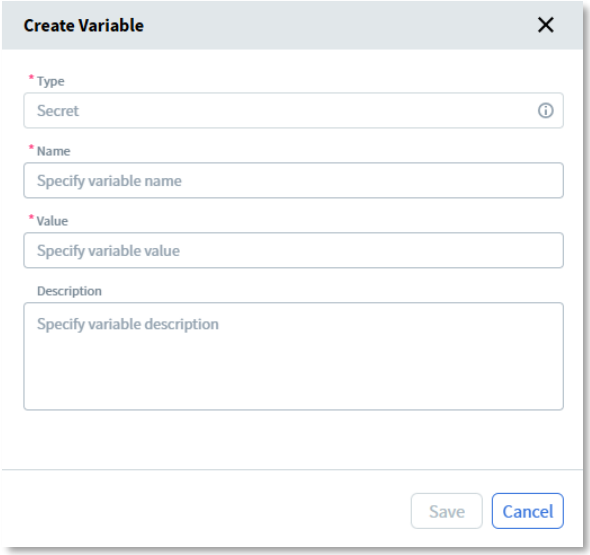

Figure 2: Create Variable Window.

Below are the environment variables that need to be set within Vantage for the integration with ResistantAI to work, all highlighted values need to be obtained from ResistantAI:

Variable 1:

- Name: ResAI ID
- Value: **Obtain from ResistantAI**
- Description: The Client ID needed to access the API

Variable 2:

- Name: ResAI Secret
- Value: **Obtain from ResistantAI**
- Description: The Secret needed to access the API

Variable 3:

- Name: ResAI URL
- Value: https://[TenantName].documents.resistant.ai/ui/analysis/[subid]
- Description: The standard link for ResistantAI's user interface. This requires your specific tenant in the URL which needs to be obtained from ResistantAI.

Variable 4

- Name: ResAI Token Url
- Value: https://eu.id.resistant.ai/oauth2/aus2un1hkrKhPjir4417/v1/token
- Description: The API endpoint for authentication to request an access token.

Variable 5:

- Name: ResAI Sub Url
- Value: https://api.documents.resistant.ai/v2/submission
- Description: The API endpoint for submissions to upload the document to.

ResistantAI's documentation for their API can be found here:
https://documents.resistant.ai/docs/v2.html

ABBYY Vantage Skill Fields

Create a Document Skill

To successfully process documents through Vantage and the integration with ResistantAI a number of fields need to be pre-set in the document extraction skill(s). These fields are used to write specific values and parameters needed to perform the integration.

In this example we will create a new document skill to set these fields, however this same process can be followed for modifying an existing skill.

To start, navigate to your **Skill Catalog (1)** in the Vantage Tenant Administration interface, select **Create (2)** from the menu and then select **Document Skill (3)**:

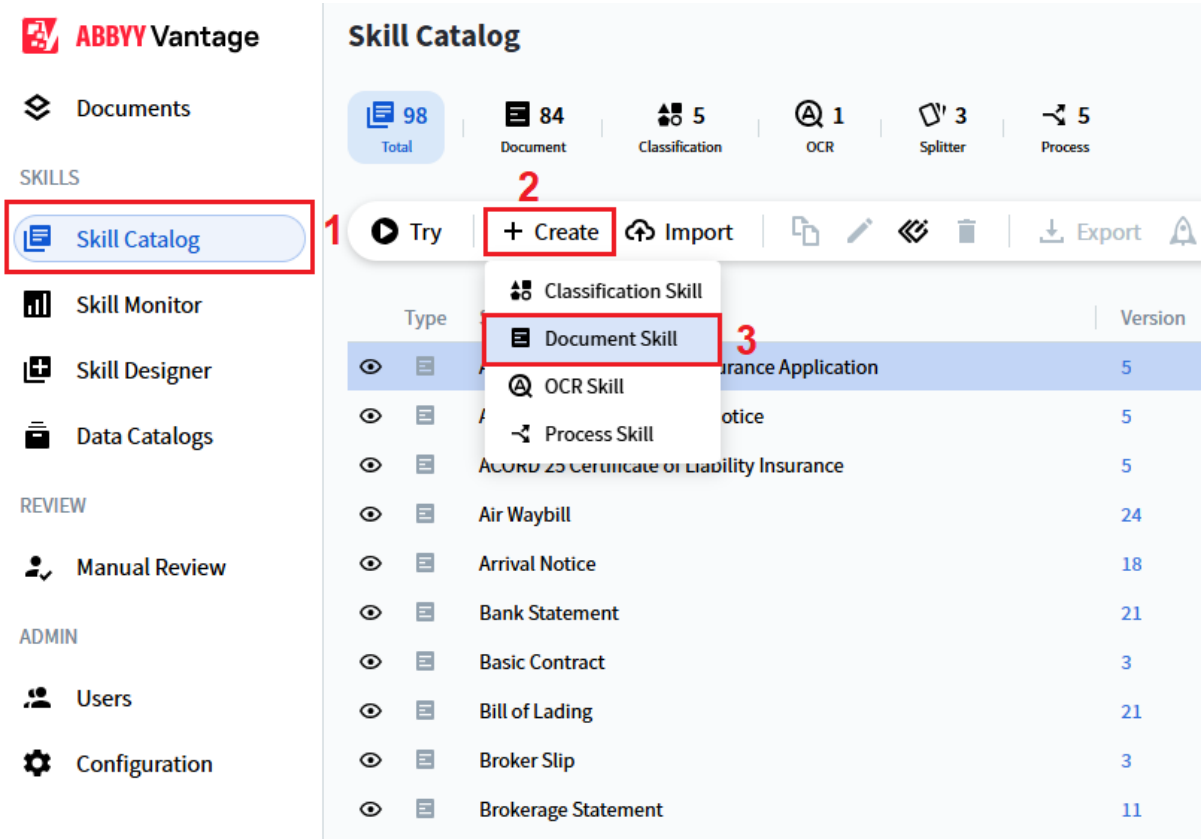

Figure 3: Skill Catalog Interface

After selecting the Document Skill icon, the following window will appear: (see next page)

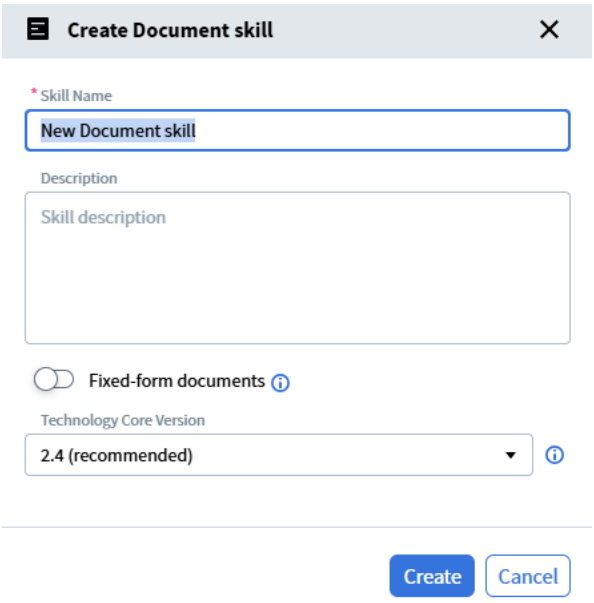

Figure 4: Create Document Skill Window

Give this new skill a name, in this example we will use the name **Demo ResistantAI Skill**, then select **Create** to load the next window:

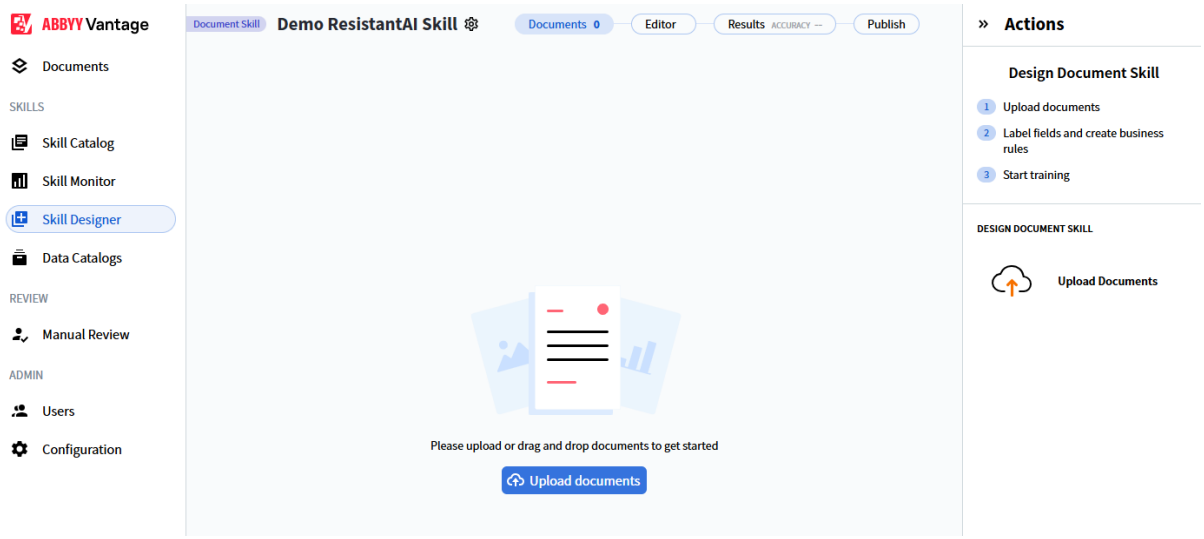

Figure 5: Skill Designer Documents Window

We now need to upload some samples to this document skill so that we can define the fields, a single document sample is sufficient to perform the next actions. Select **Upload Documents** from the right-hand pane and navigate to your document sample. The menu on the right-hand pane will change once the upload is complete, select the option of **Label Fields and Create Business Rules**:

(see next page).

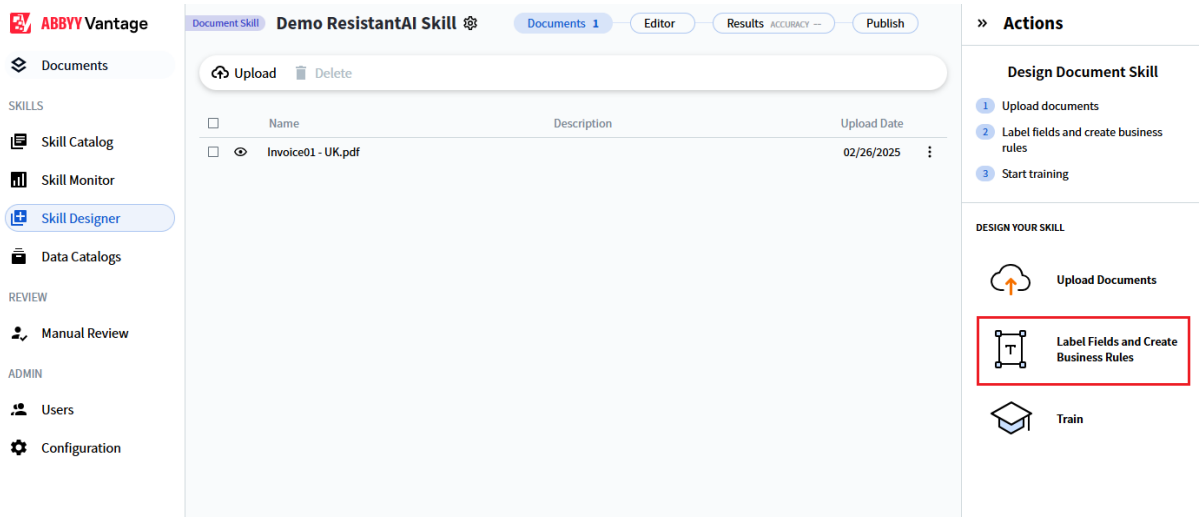

Figure 6: Skill Designer Documents Window - After Document Upload

The interface will change and move into the editor (see below), now we are ready to define the field structure needed for the integration with ResistantAI:

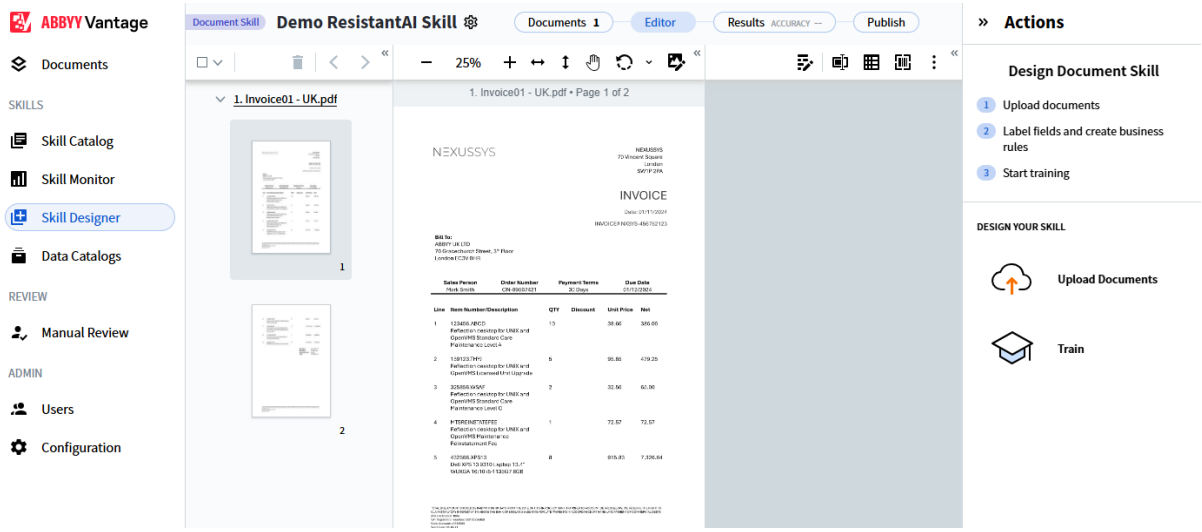

Figure 7: Skill Designer Editor Interface

Create Required Field Structure

The fields that are required for the process can now be created in this skill; to create them we will be using the field creation icons from the top right pane:

- Add Field Icon
- Add Group Icon Group

First select the **Group** icon to create a new group, double left-click on the name of the group and rename this to **Fraud Check**, press enter on the keyboard to confirm the rename**:**

(see next page)

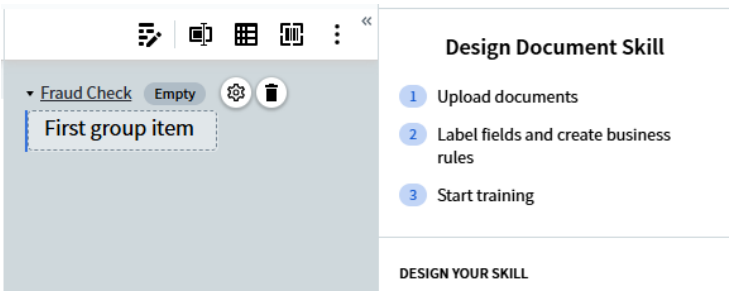

Figure 8: Group Field Created and Renamed

Click on the text **First group item** so that it is selected (the blue highlight left of the field text indicates it is selected) as we need to create fields inside the group, if this is not selected then the fields will be created outside of the group incorrectly.

Select the **Add Field Icon** twice to create two new fields inside the group:

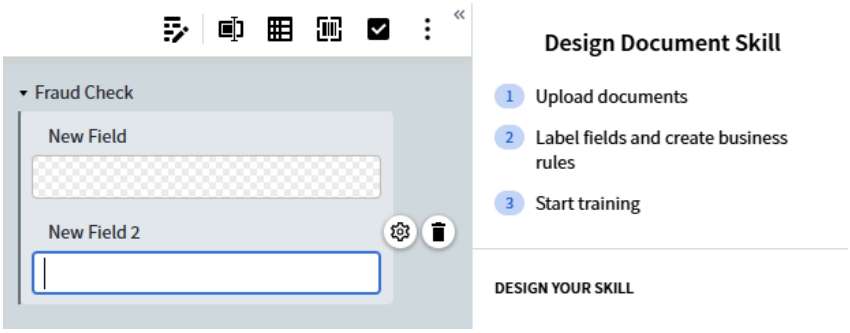

Figure 9: New Fields Inside the Group

We now need to rename each field, to do this double left-click on each field name and use the corresponding field names, confirm each rename with the enter key on the keyboard:

- Score
- Analysis Report URL

After this is completed, the fields should appear like this:

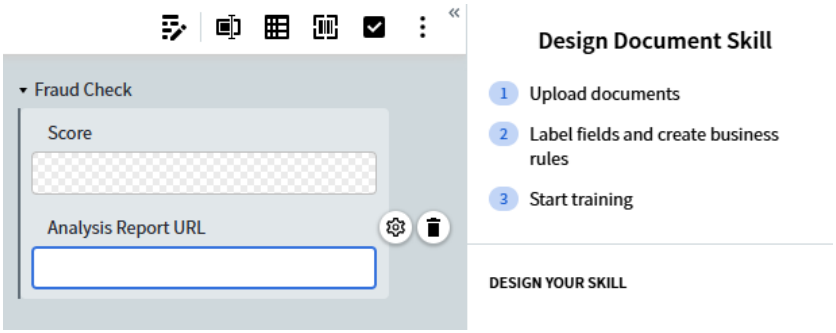

Figure 10: Renamed Fields Inside the Group

Now create a new group inside of the **Fraud Check** group, complete this by selecting the **Add Group** icon, rename this new group **Risk Indicators**:

(see next page)

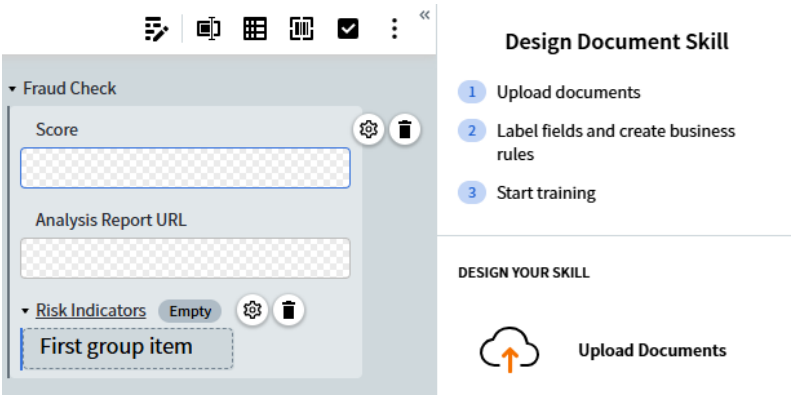

Figure 11: Risk Indicators Group Created

Inside of the Risk Indicators group, create a new field called **Indicator**, use the **Add Field** icon to complete this, then double-left click on the name of the field to rename it and press enter on the keyboard to confirm the change:

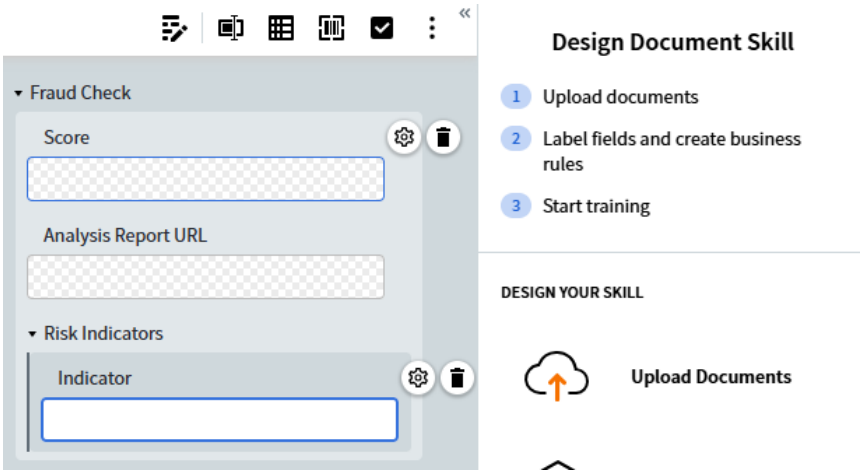

Figure 12: Indicator Field Created

Adjust Field Settings

The **Indicator** field needs a setting changed on it, this is to allow the field to be repeated so that each indicator can be written to it's own instance in the group. To make this change hover the mouse over the field so the cog icon is visible (refer to Figure 12 above). Select the cog icon with the left-mouse button to load the following window and then select **Advanced**:

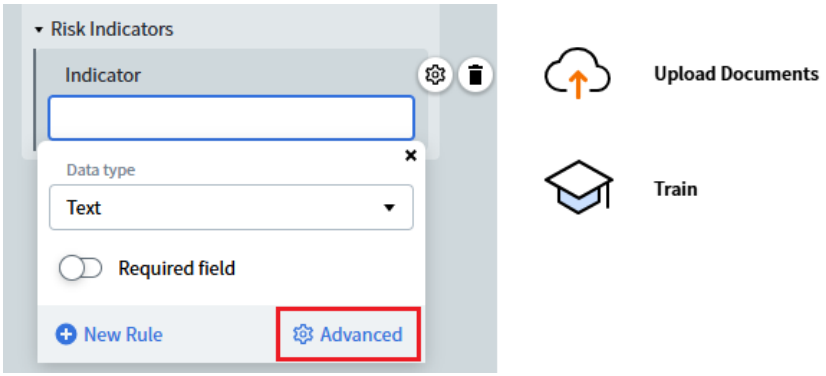

Figure 13: Field Menu Window

The following window will load, from this window select the toggle for **Allow Multiple Items**, this will enable the functionality needed for this process:

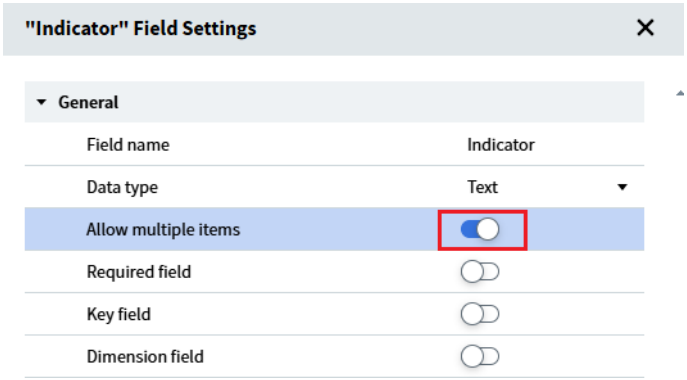

Figure 14: Enable Allow multiple items

Click on **Save** at the bottom of the window to save changes to this field setting, the field on the Editor interface will now appear slightly different with the **Add 'Indicator' item** visible:

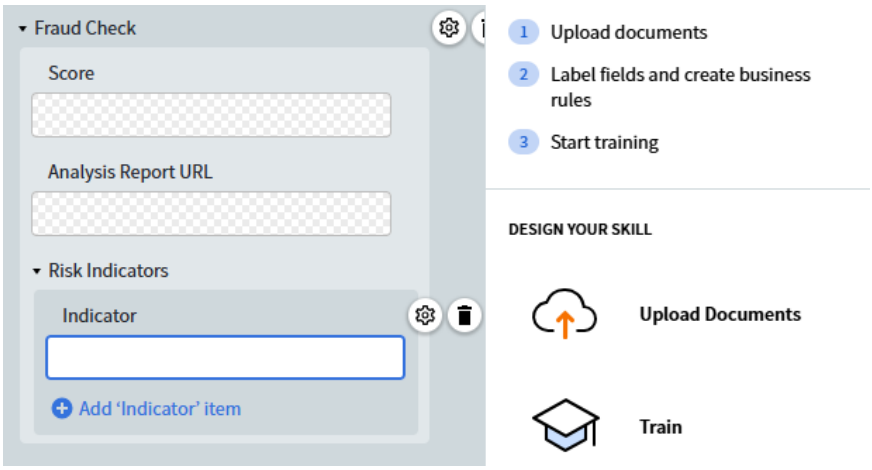

Figure 15: Field Interface Update

All fields are now setup for the document skill.

Publish the Document Skill

As this guide is only covering the integration with ResistantAI, no training of the document skill will be covered. For now, select **Publish** from the top of the designer interface:

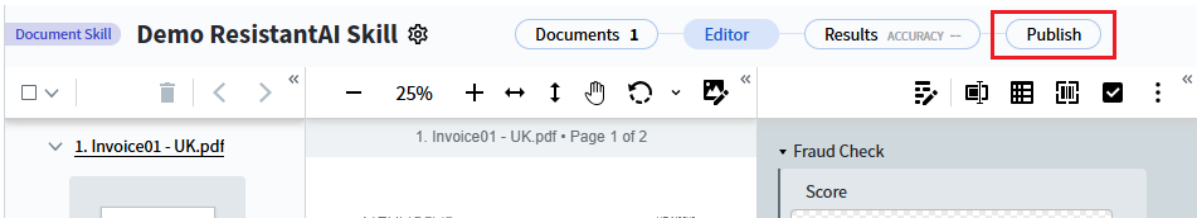

Figure 16: Skill Designer Interface - Publish

Then select **Publish** from the right-hand pane to make the skill available for use with a Process Skill, which will be covered next:

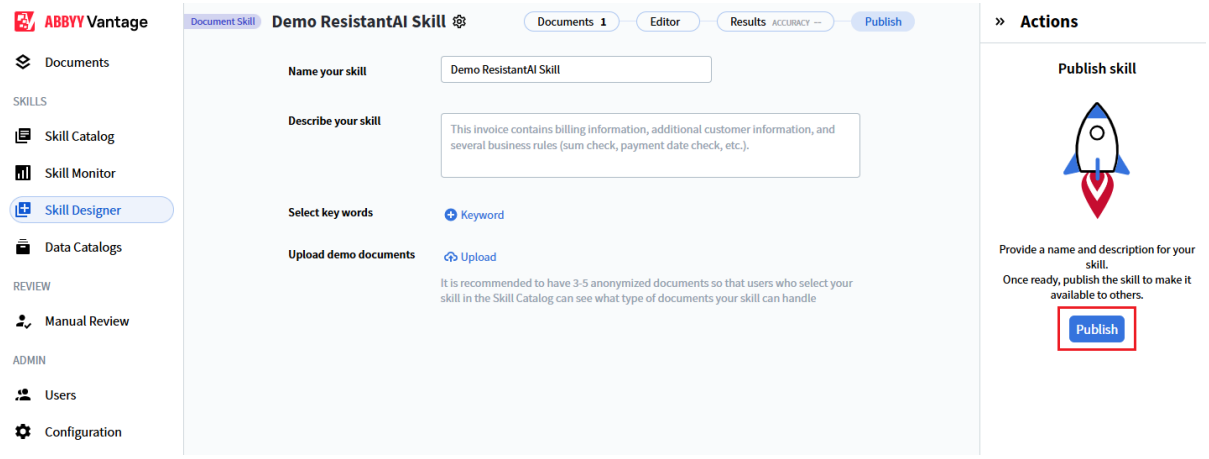

Figure 17: Skill Designer Interface – Publish

Create a Process Skill

To create a Process Skill, go back to the **Skill Catalog (1)**, select **Create (2)**, then select **Process Skill (3):**

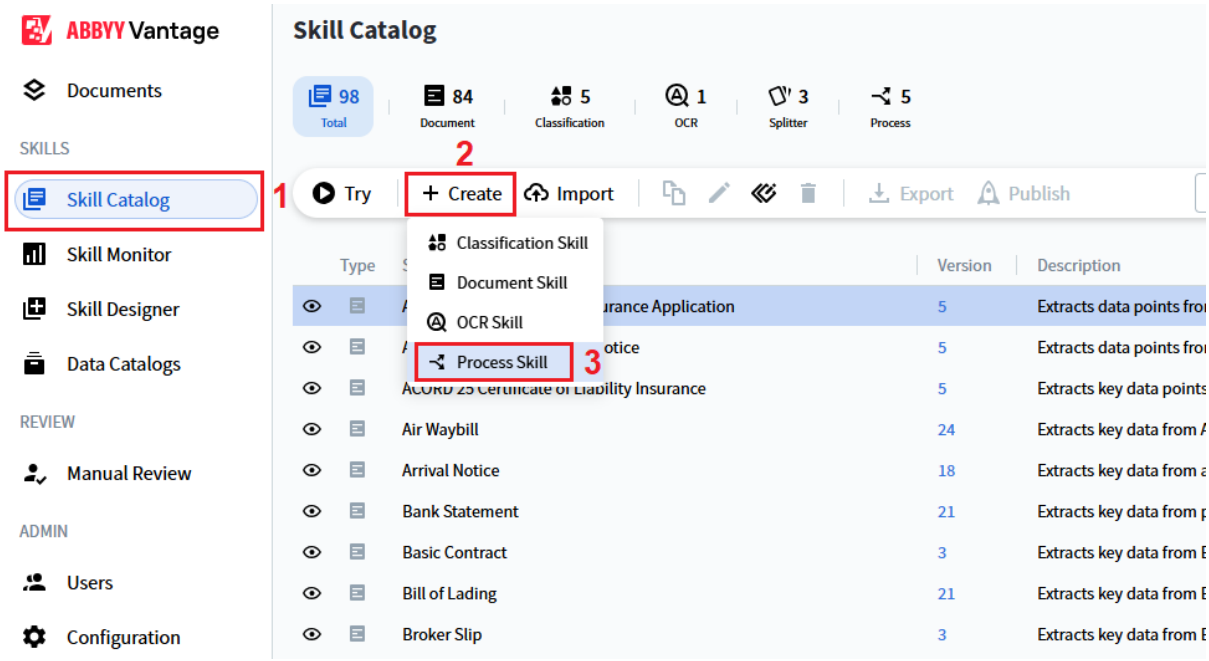

Figure 18: Skill Catalog Interface

In the window that loads up, name your process skill, then select **Create** from the bottom of the window:

(see next page)

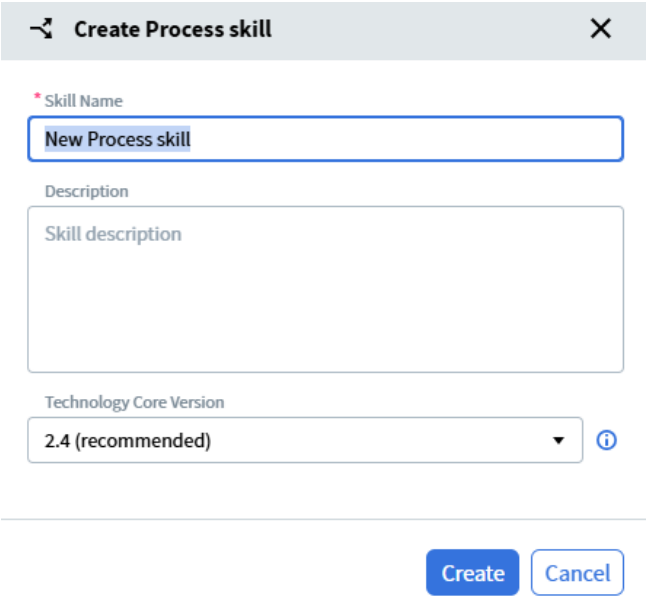

Figure 19: Create Process Skill Window

In this example we have renamed the process skill 'Fraud Integration Process Skill'.

Using the activities pane in the process skill designer, create a process flow in the skill designer window, to start select **Input (1)**, then click anywhere in the skill designer interface to place the I**nput (2)**:

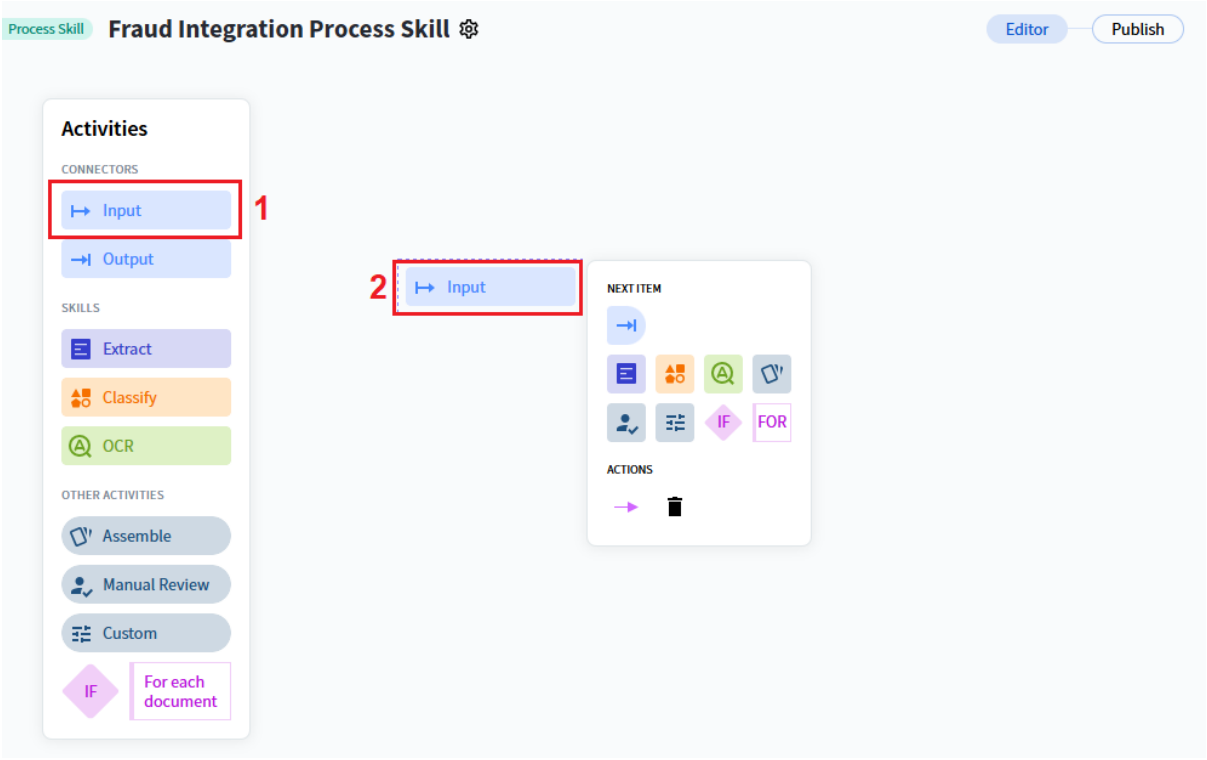

Figure 20: Process Skill Designer Interface

From the sub menu window that appears whilst the input is selected, click on the icons for the following activities so that we have a workflow connected like the example below:

Input>Extract>Custom Activity>Manual Review>Output

Figure 21: Process flow inside Process Skill

Select the **Extract** stage **(1)** and click **Add Skill (2)** from the right-hand pane to add the Document Skill that was created earlier (see Create a Document Skill):

Figure 22: Process flow Add Skill

Now we need to populate the custom activity with the script required to complete the integration.

Custom Activity

Select the Custom Activity stage from the process flow in the skill designer interface, then select **Settings** from the right-hand pane to load the JavaScript interface:

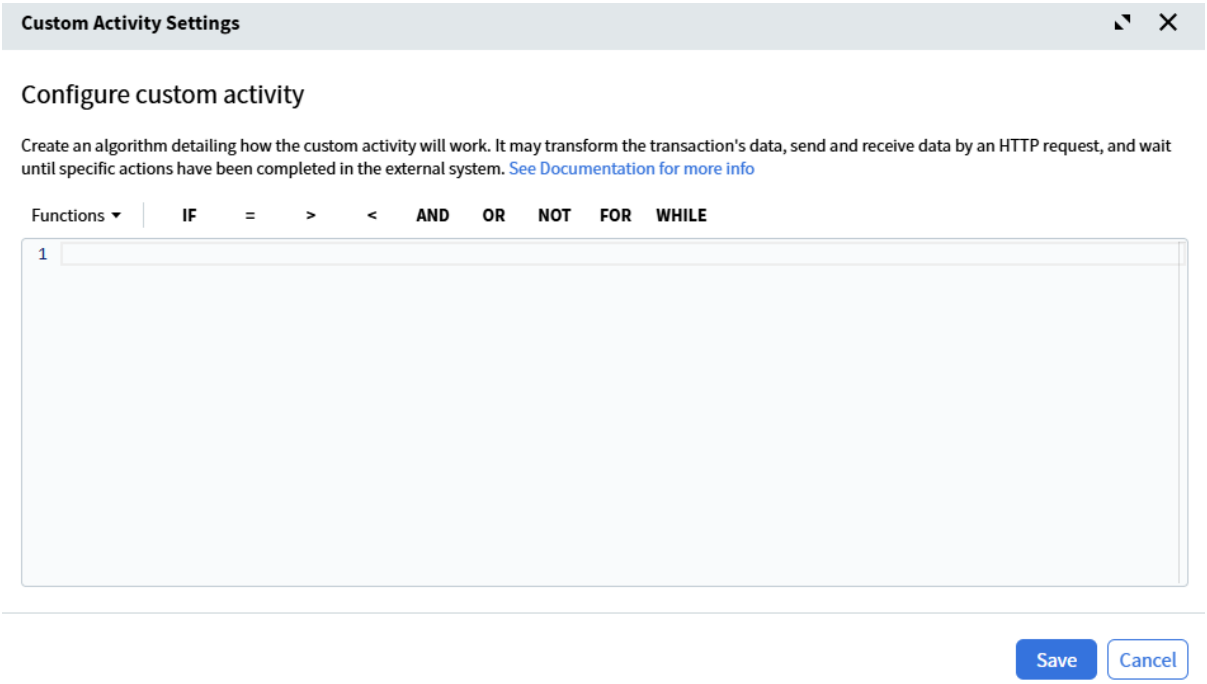

Figure 23: Custom Activity JavaScript Interface

Copy the whole script from Appendix A – Custom Activity Script and paste it into this window, click **Save** on the interface to save changes.

Then click Publish on the process skill to move to the publish interface, then select Publish again to publish the process skill and make this available for testing.

Summary

Provided that the environment variables are named correctly, and the fields are setup correctly in the document skill the process flow will work successfully.

Several logs will be written to the skill monitor for the transaction to confirm steps in the process, or if the custom activity fails to process the document.

The custom activity step will process all documents in the transaction through to ResistantAI's platform, this happens in a linear mode for each transaction.

Once all documents have processed through then the task to review the documents will appear at the Manual Review stage.

Appendix A – Custom Activity Script

```
var accessToken = null;


//Defined flow of Custom Activity
function RunFlow(){


 //Run the flow for each document in the transaction.
 for (var doc of Context.Transaction.Documents) {


  var token = GetBearerToken();
  var returnedJSON = CreateSubmission(token);
  UploadDocument(doc, returnedJSON.upload_url);
  var subId = returnedJSON.submission_id;


  //Return the result from ResistantAI
  var result = GetStatus(token, subId);


  // try until 10 times get the result.
   for (var i=1; i<10; i++) {


    if (result[0]==200) {


     WriteToFields(doc, result[0], result[1], subId);
     break;
    }
    else {


     TakeSometime();//This pauses the workflow for 10 seconds. This can be adjusted.
     result = GetStatus(token, subId); //Return the result.
     Context.LogMessage("Result" + i.toString() + ": " + result.toString()); // Log the result to the transaction log.
```

```
    }
   }
  }
 }

//Upload the document to the submission ID created
function UploadDocument(doc,url){

 var exports = doc.SourceFiles[0];

 var uploadRequest = Context.CreateHttpRequest();
 uploadRequest.Method = "PUT";
 uploadRequest.Url = url;
 uploadRequest.SetFileContent(exports, "application/octet-stream");

 uploadRequest.Send();
}

//Returns a bearer token from ResistantAI API.
function GetBearerToken(){

 // Prepare the key-value body for the authorization request
 var authDataContent = {};
 authDataContent.grant_type = "client_credentials";
 authDataContent.scope = "submissions.read submissions.write";

 // Pass the ID and secret to access ResAI's API.
 authDataContent.client_id = Context.GetSecret("ResAI ID");
 authDataContent.client_secret = Context.GetSecret("ResAI Secret");

 // create the request and send the data
```

```
var request = Context.CreateHttpRequest();
request.Url = Context.GetSecret("ResAI Token Url");
request.Method = "POST";
request.SetHeader("accept","application/json");

// create content data for the authorization request
request.SetUrlFormEncodedContent(authDataContent);

//Send the request, log the error to the transaction log if there is one.
try{

 request.Send();
}
catch(err){

 Context.LogMessage(err);
}

var authResponseObject = JSON.parse(request.ResponseText);

/* The JSON file returned will be in the following format:
{
 "access_token": "eyJhbG[...]1LQ",
 "token_type": "Bearer",
 "expires_in": 3600,
 "scope": "submissions.read submissions.write"
}
*/

//Retrieve the access token from the JSON
accessToken = authResponseObject.access_token;
```

```
  return accessToken;
 }

 //Write the Risk indicators in the JSON response to the repeatable title field
 //in the indicators group in the document.
 function WriteIndicators(doc, result, subId){

  //Create a new instance of the Risk Indicators group
  doc.GetField("Fraud Check/Risk Indicators").AddInstance();

    //Loop through each group of indicators and write them to the fields in a new group for each one
    result.indicators.forEach(function (elem){
      if(elem.type == "RISK"){
        var newGroup = doc.GetField("Fraud Check/Risk Indicators/Indicator").AddInstance();
        newGroup.Value = elem.title;
      }
  })

  var urlField = doc.GetField("Fraud Check/Analysis Report URL");
  var urlParam = Context.GetSecret("ResAI URL");
  Context.LogMessage("ResAI URL" + urlParam);
  var urlValue = urlParam.replace("[subid]", subId);

  urlField.Value = urlValue;
 }

//Call this external API to pause the flow in Vantage
function TakeSometime(){
  try {
```

```
    var resultRequest = Context.CreateHttpRequest();
    resultRequest.ThrowExceptionOnFailed = false;
    resultRequest.Url = "https://httpbin.org/delay/10"; //delay of 10 seconds
    resultRequest.Method = "GET";
    resultRequest.Send();
    return resultRequest.ResponseText;
   } catch {}
}

//Call various functions to write the values to the required fields in the documents.
function WriteToFields(doc, status, result, subId){

  var obj = JSON.parse(result); //Get the json result and parse it
  doc.GetField("Fraud Check/Score").Value = obj.score; //write the score value to the Score field
  WriteIndicators(doc, obj, subId); //Call the WriteIndicators function to write all indicators to the field group.

}

//Creates a GET request to return JSON response from Resistant AI's API based on the subId
function GetStatus(token, subId){

  var resultRequest = Context.CreateHttpRequest();
  resultRequest.ThrowExceptionOnFailed = false;
  resultRequest.Url = `https://api.documents.resistant.ai/v2/submission/${subId}/fraud`;
  resultRequest.Method = "GET";

  resultRequest.AuthScheme = "Bearer";
  resultRequest.AuthToken = token;
  resultRequest.SetHeader("Accept","application/json");
```

```
  //send the request to get the JSON
  resultRequest.Send();

  return [resultRequest.Status , resultRequest.ResponseText];

}

//Create a submission so we can upload the document to ResistantAI
function CreateSubmission(token){

  var request = Context.CreateHttpRequest();
  request.Method = "POST";
  request.Url = Context.GetSecret("ResAI Sub Url");
  request.AuthScheme = "Bearer";
  request.AuthToken = token;
  request.SetHeader("Content-Type","application/json");

request.SetStringContent(JSON.stringify({query_id:"123456",pipeline_configuration:"FRAUD_ONLY",enable_decision:"false",enable_submission_characteristics:"false"}));

  request.Send();
  var responseObject = JSON.parse(request.ResponseText);

  return responseObject;
}

RunFlow();
```