ABBYY Vantage

Integration with Finovox Fraud Detection Platform

Developed by: Deepak Goyal

Contents

About ABBYY Vantage Integration to Finovox 3
System Requirements 4
  Prerequisites 4
    Environment Variables 4
    Security Considerations 5
  ABBYY Vantage Skill Fields 5
    Create a Document Skill 5
    Create Required Field Structure 7
    Publish the Document Skill 8
  Create a Process Skill 9
  Custom Activity 12
  Troubleshooting 12
  Best Practices 12
Summary 13
Appendix A – Custom Activity Script 14

About ABBYY Vantage Integration to Finovox

This integration guide describes how to integrate ABBYY Vantage with Finovox'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 Finovox 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 Finovox 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.

Use Cases

- The integration of ABBYY Vantage with Finovox enables several real-world fraud detection scenarios:
- Invoice validation – Detects forged or manipulated invoices before payment.
- Contract analysis – Highlights fraudulent or unauthorized edits in legal documents.
- KYC document verification – Identifies inconsistencies in identity documents during onboarding.
- Vendor onboarding – Flags risky or blacklisted entities based on supporting documents.

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.

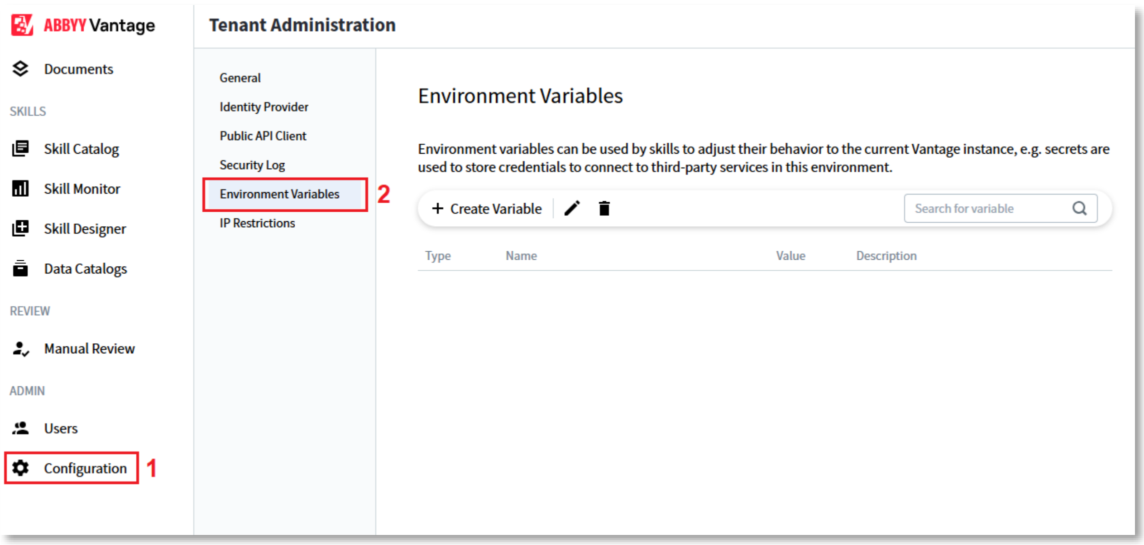
You will also require a valid account for the Finovox platform for API Key.

Prerequisites

Environment Variables

For the process to run successfully 1 environment variable is required, this will be configured within the Vantage administration interface. From the side panel in the main tenant administration interface go to:

1. Configuration
2. Environment Variables:



Environment Variables

Environment variables can be used by skills to adjust their behavior to the current Vantage instance, e.g. secrets are used to store credentials to connect to third-party services in this environment.

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:

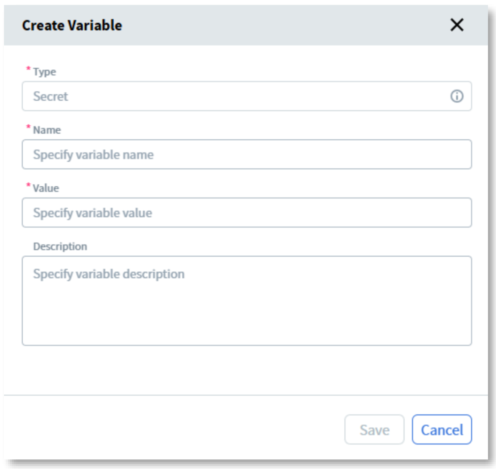


Figure 2: Create Variable Window.

Below is the environment variable that need to be set within Vantage for the integration with Finovox to work, all highlighted values need to be obtained from Finovox:

Variable 1:

- Name: Finovox_API_Key
- Value: **Obtain from Finovox**
- Description: The API Key needed to access the API

Finovox's documentation for their API can be found here: [Finovox API | Finovox Documentation]

Security Considerations

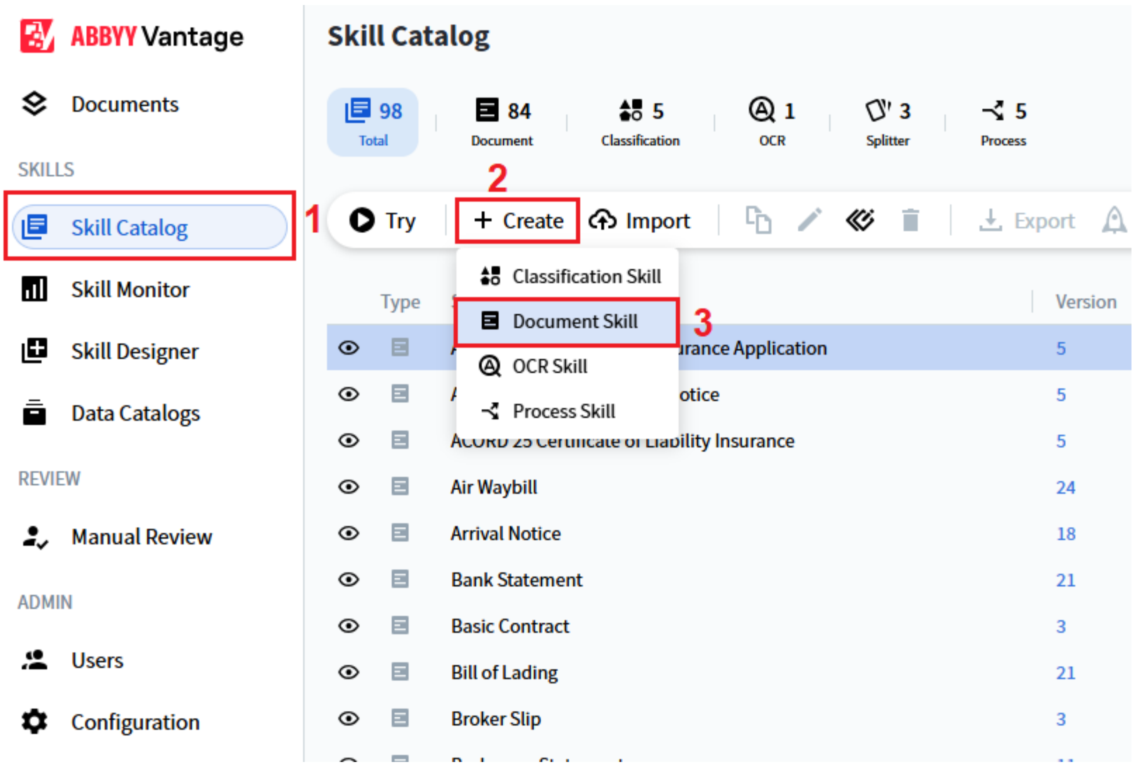
- API keys must always be stored in Vantage Secrets, never hard-coded in scripts.
- Ensure all data transfers are secured via HTTPS.
- Evaluate compliance and data privacy requirements before sending PII outside the tenant.
- Restrict tenant access to Environment Variable configuration to authorized admins only.

ABBYY Vantage Skill Fields

Create a Document Skill

To successfully process documents through Vantage and the integration with Finovox 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.

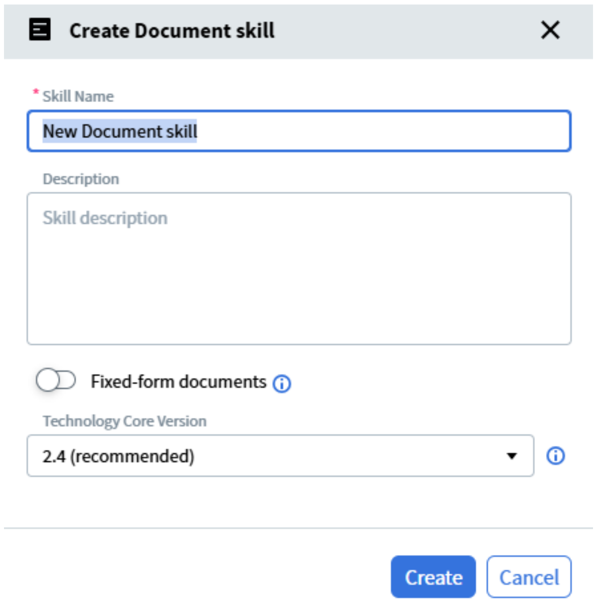
To start, navigate to your **Skill Catalog (1)** in the Vantage Tenant Administration interface, select **Create (2)** from the menu and then select **Document Skill (3)**:



Name	Version
...rance Application	5
...otice	5
ACORD 25 Certificate of Liability Insurance	5
Air Waybill	24
Arrival Notice	18
Bank Statement	21
Basic Contract	3
Bill of Lading	21
Broker Slip	3

Figure 3: Skill Catalog Interface

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


Create Document skill

* Skill Name

New Document skill

Description

Skill description

Fixed-form documents

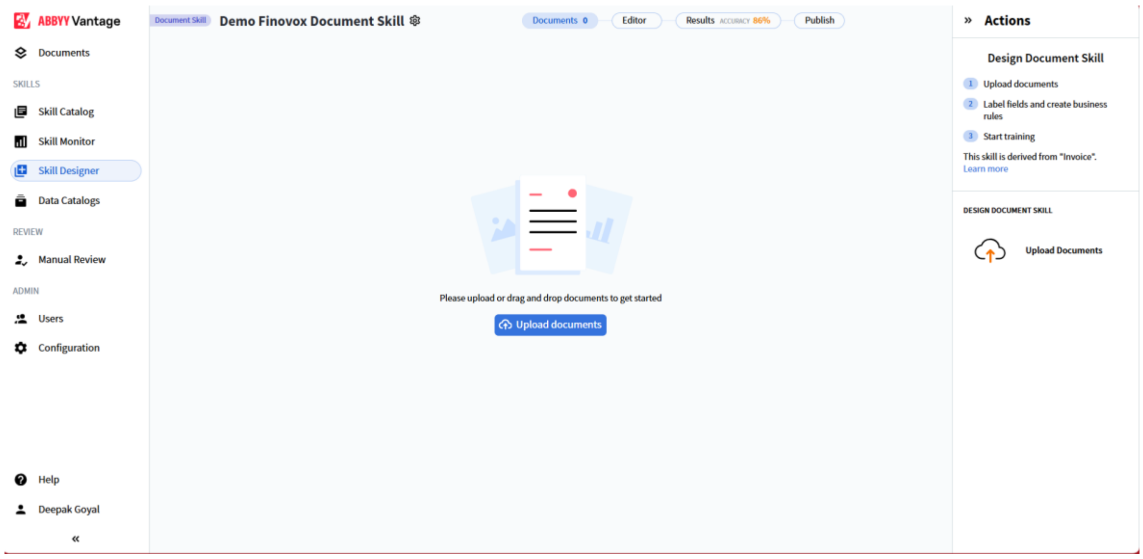
Technology Core Version

2.4 (recommended)

Create Cancel

Figure 4: Create Document Skill Window

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



Design Document Skill

1 Upload documents

2 Label fields and create business rules

3 Start training

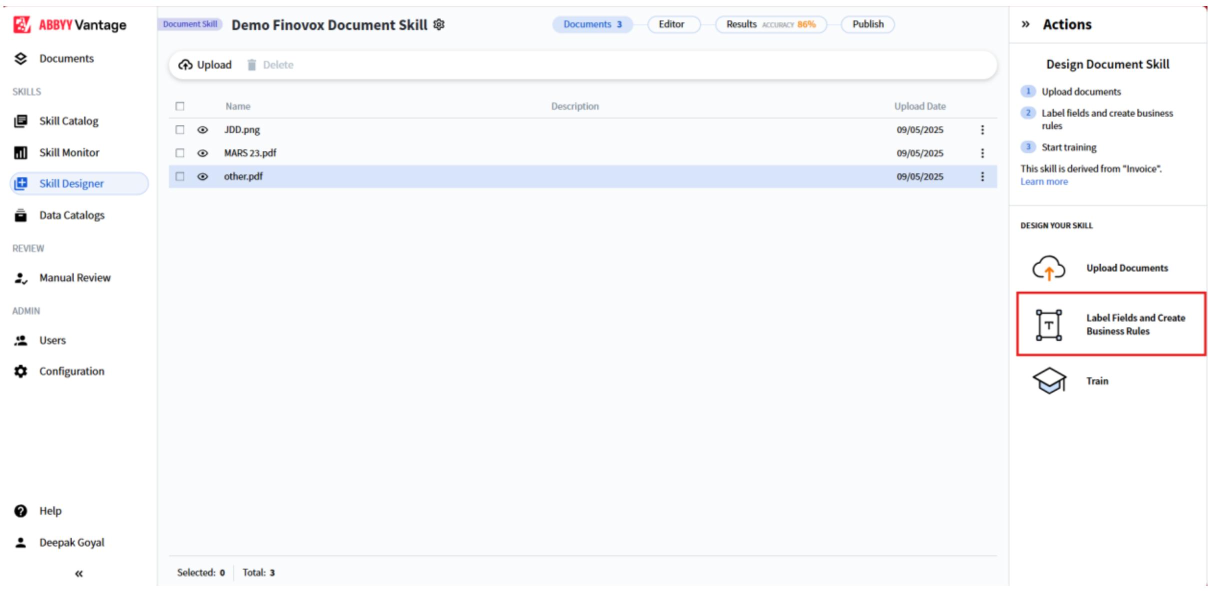
This skill is derived from "Invoice". Learn more

Please upload or drag and drop documents to get started

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).

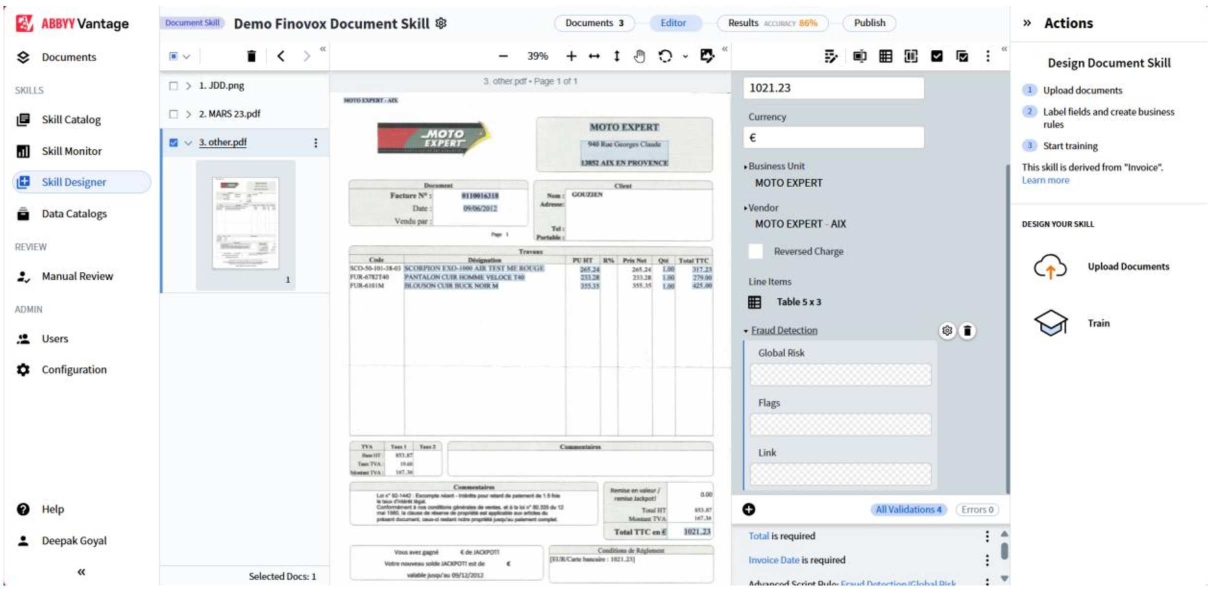


Name	Description	Upload Date
JDD.png		09/05/2025
MARS 23.pdf		09/05/2025
other.pdf		09/05/2025

Selected: 0 Total: 3

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 Finovox:



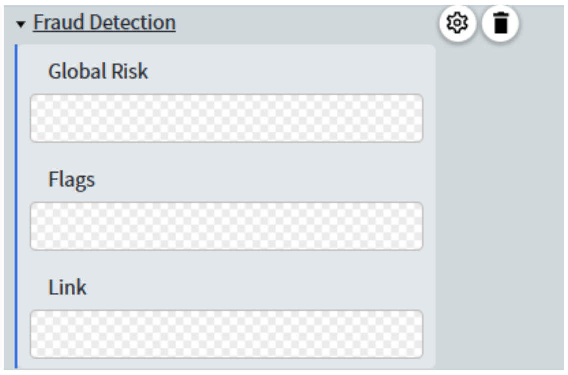
1021.23
Currency
€
Business Unit
MOTO EXPERT
Vendor
MOTO EXPERT - AIX
Reversed Charge
Line Items
Table 5 x 3
Fraud Detection
Global Risk
Flags
Link
Total is required
Invoice Date is required

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



Publish the Document Skill

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

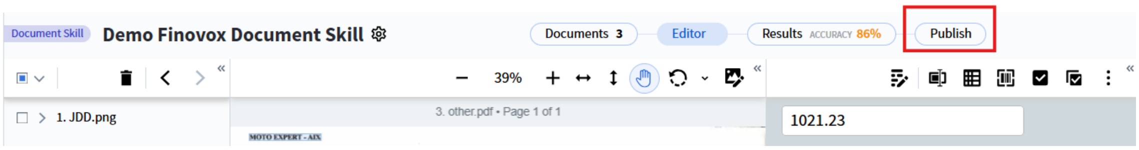
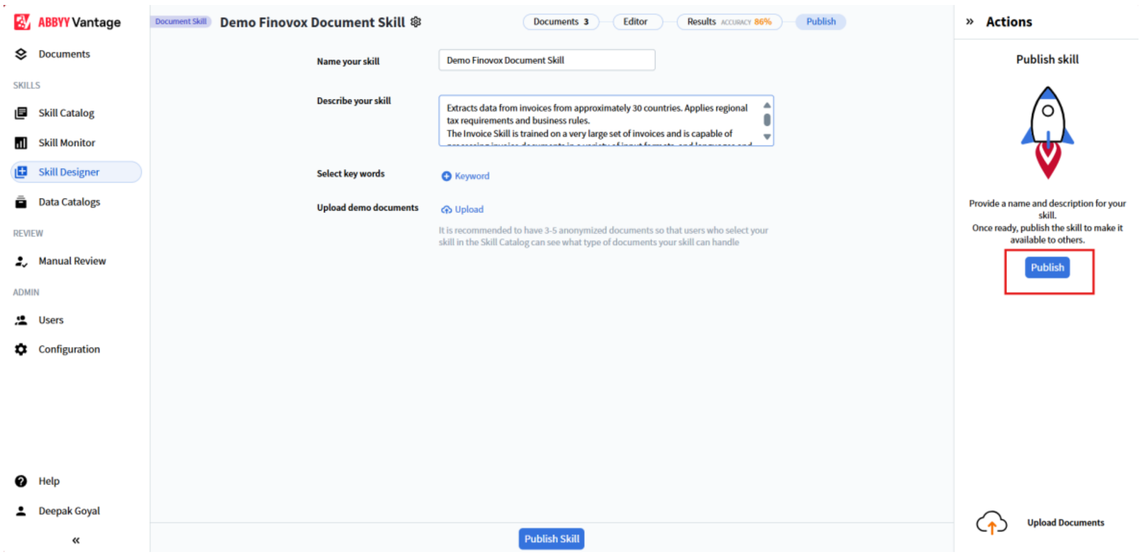


Figure 8: Skill Designer Interface - Publish

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



Name your skill: Demo Finovox Document Skill

Describe your skill: Extracts data from invoices from approximately 30 countries. Applies regional tax requirements and business rules.
The Invoice Skill is trained on a very large set of invoices and is capable of

Select key words

Upload demo documents

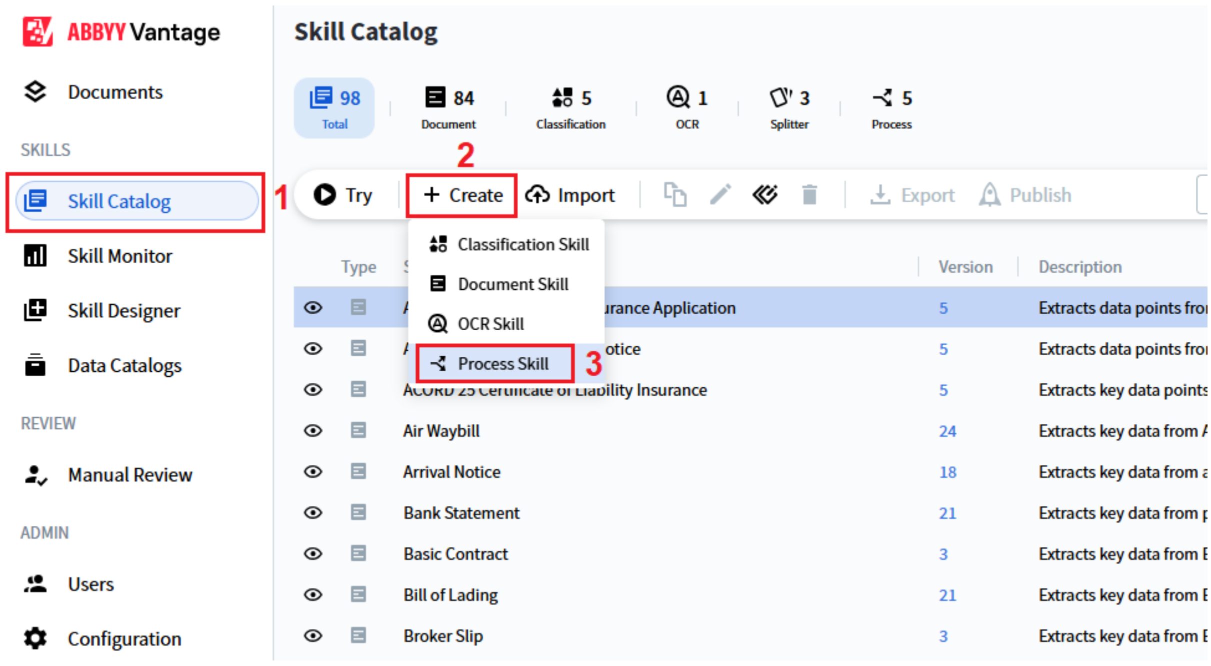
It is recommended to have 3-5 anonymized documents so that users who select your skill in the Skill Catalog can see what type of documents your skill can handle

Provide a name and description for your skill.
Once ready, publish the skill to make it available to others.

Figure 9: 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):**

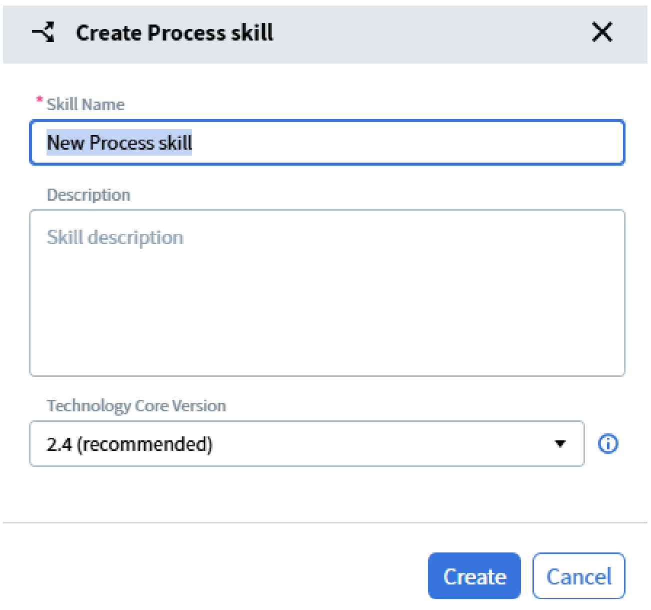


Type		Version	Description
	A urance Application	5	Extracts data points fro
	A otice	5	Extracts data points fro
	ACORD 25 Certificate of Liability Insurance	5	Extracts key data points
	Air Waybill	24	Extracts key data from A
	Arrival Notice	18	Extracts key data from a
	Bank Statement	21	Extracts key data from p
	Basic Contract	3	Extracts key data from B
	Bill of Lading	21	Extracts key data from B
	Broker Slip	3	Extracts key data from B

Figure 10: 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)



* Skill Name: New Process skill

Description: Skill description

Technology Core Version: 2.4 (recommended)

Figure 11: 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 **Input (2)**:

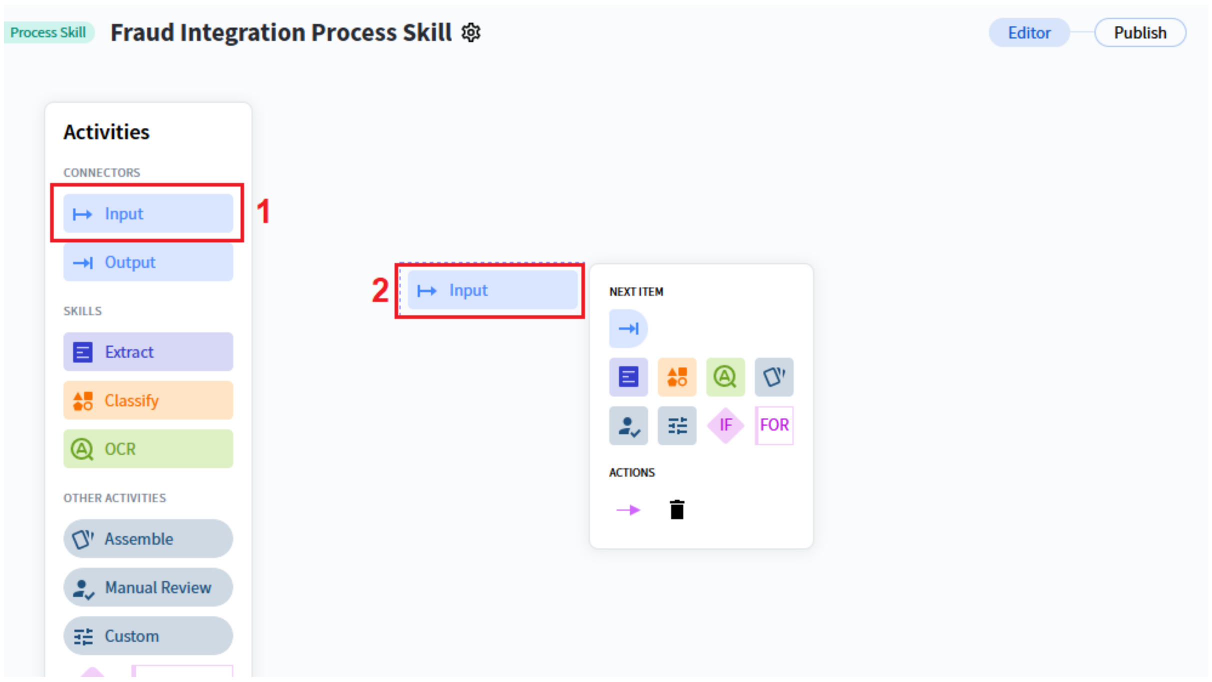


Figure 12: Process Skill Designer Interface

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

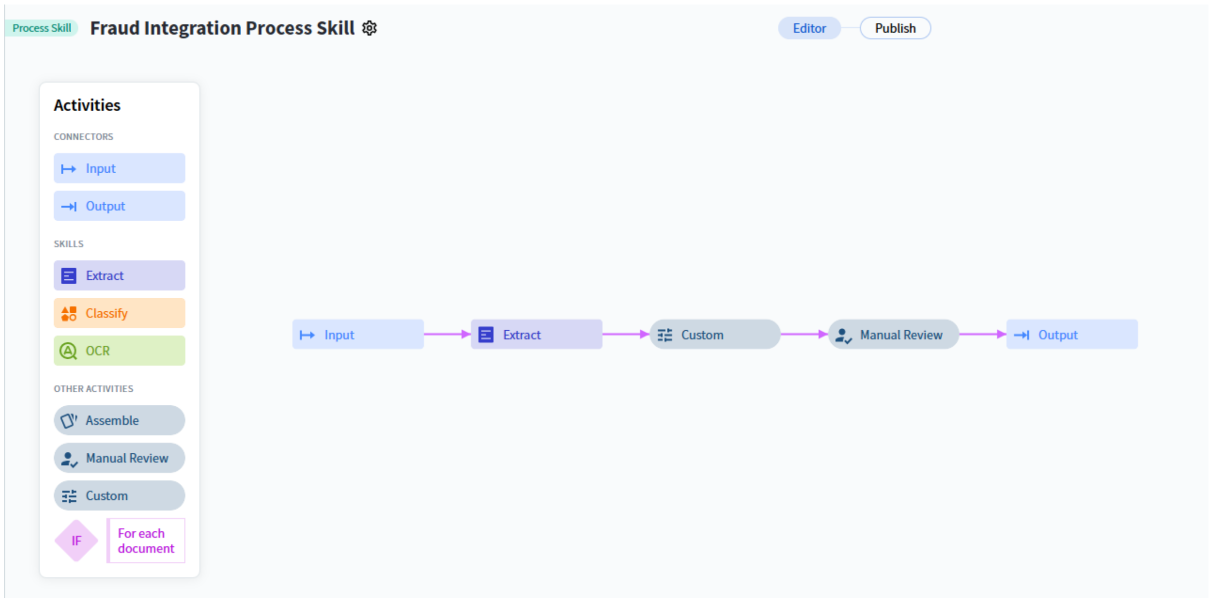


Figure 13: Process flow inside Process Skill

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):

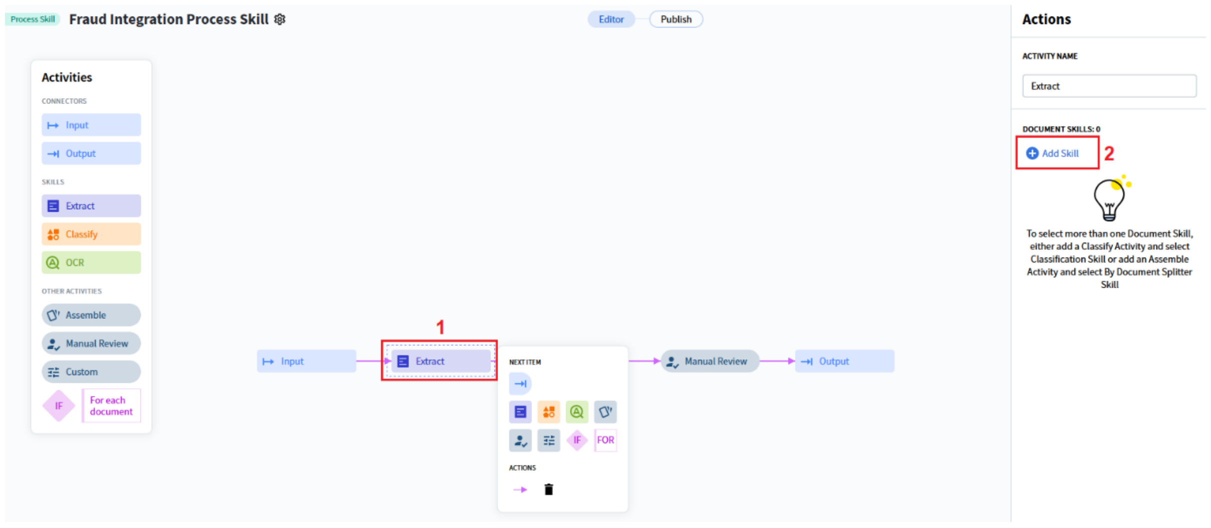
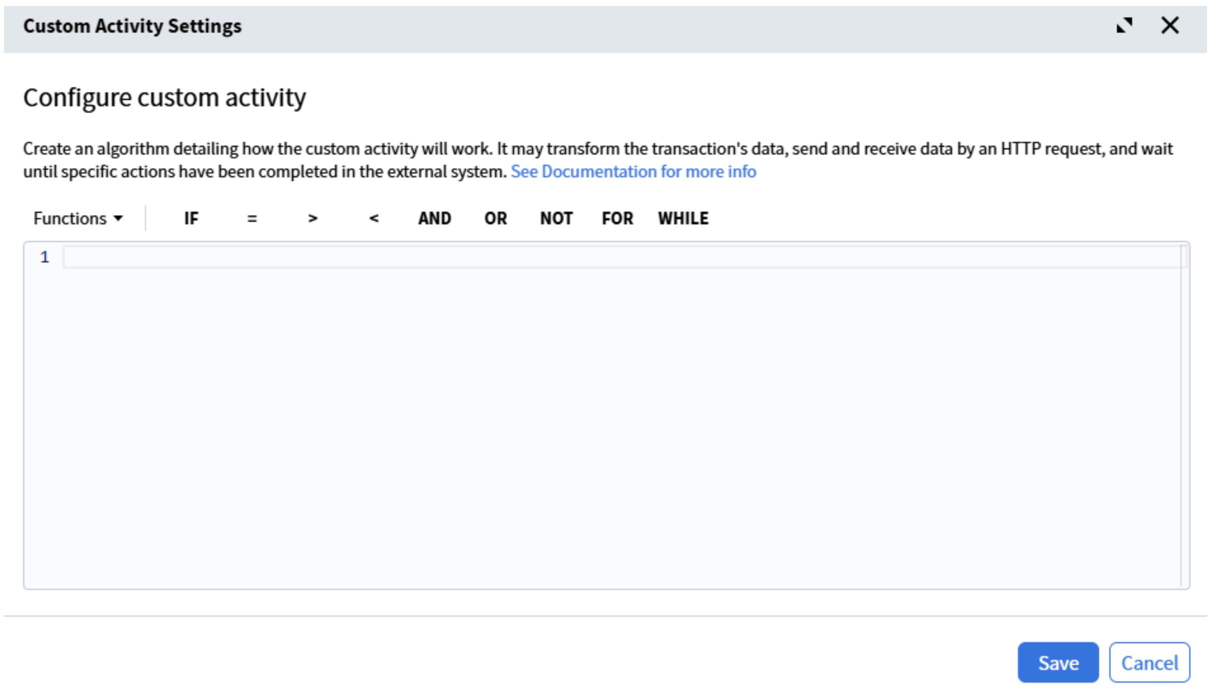


Figure 14: Process flow Add Skill

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:


Custom Activity Settings

Configure custom activity

Create an algorithm detailing how the custom activity will work. It may transform the transaction's data, send and receive data by an HTTP request, and wait until specific actions have been completed in the external system. See Documentation for more info

Figure 15: 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.

Troubleshooting

Common issues and resolutions:

- Missing API key – Ensure the environment variable Finovox_API_Key is created and populated.
- Invalid JSON – Confirm the Finovox API response; check for proxy or firewall interference.
- Fields not updating – Ensure required output fields exist in the Document Skill and are writable.
- Polling timeout – Increase MAX_POLLS or verify that Finovox jobs are completing successfully.

Best Practices

- Place the custom activity after extraction and before Manual Review for early fraud detection.
- Limit payload size by sending only necessary documents to Finovox.
- Monitor the Skill Monitor logs regularly for anomalies or failed transactions.
- Keep the integration updated in line with the latest Finovox API changes.

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 Finovox'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

```
// ===== Finovox Async – Document-scope (ABBYY Vantage Custom Activity) =====
// Uses MultipartFormDataRequest for /analyse and HttpRequest for /retrieve

var FINOVOX_ANALYSE_URL  = "https://api-v2.finovox.com/analyse";
var FINOVOX_RETRIEVE_URL = "https://api-v2.finovox.com/retrieve";
var MAX_POLLS = 20;

// Pause helper per your convention
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 {}
}

function RunFlow() {
  // --- API key
  var apiKey = Context.GetSecret("Finovox_API_Key");
  if (!apiKey || ("" + apiKey).trim() === "") {
    throw new Error('Missing secret: "Finovox API Key"');
  }
  apiKey = ("" + apiKey).trim();

  // --- Current document & first source file
  var doc = Context.Transaction.Documents[0];
  var srcFile = doc.SourceFiles[0];

  // --- 1) /analyse -> retrieve_token
  var retrieveToken = AnalyseAndGetToken(apiKey, srcFile);

  // --- 2) Poll /retrieve -> result
  var result = RetrieveUntilReady(apiKey, retrieveToken);

  // --- 3) Write fields
  WriteFields(doc, result);
}

// --------- /analyse (multipart: file + payload) ----------
function AnalyseAndGetToken(apiKey, fileObj) {
  // payload must be a JSON string field named "payload"
  var payloadObj = {
    analyse_type: ["risk", "explanation", "share_link"],
    asynchronous: true
  };
  var payloadStr = JSON.stringify(payloadObj);

  var req = Context.CreateMultipartFormDataRequest();
  req.Method = "POST";
  req.Url = FINOVOX_ANALYSE_URL;
  req.SetHeader("api-key", apiKey);

  // Add JSON payload as a string part named "payload"
  req.AppendStringContent(payloadStr, "payload");

  // Add the file part named exactly "file"
  req.AppendFileContent(fileObj, "file", "application/pdf");

  req.Send();

  var respText = req.ResponseText;
  var obj;
  try { obj = JSON.parse(respText); }
  catch (e) { throw new Error("Finovox /analyse: invalid JSON. Raw: " + respText); }

  if (!obj.retrieve_token) {
    throw new Error("Finovox /analyse: missing retrieve_token. Body: " + respText);
  }
  return obj.retrieve_token;
}

// --------- /retrieve (robust polling with status checks) ----------
function RetrieveUntilReady(apiKey, retrieveToken) {
  var lastStatus = -1;
  var lastBody = "";

  for (var i = 0; i < MAX_POLLS; i++) {
    var req = Context.CreateHttpRequest();
    req.Method = "POST";
    req.Url = FINOVOX_RETRIEVE_URL;
    req.ThrowExceptionOnFailed = false;
    req.SetHeader("api-key", apiKey);
    req.SetHeader("Content-Type", "application/json");
    req.SetStringContent(JSON.stringify({ retrieve_token: retrieveToken }));

    try { req.Send(); } catch (err) { Context.LogMessage("Retrieve send error: " + err); }

    lastStatus = req.Status;
    lastBody   = req.ResponseText;

    if (lastStatus !== 200) {
      Context.LogMessage("Finovox /retrieve attempt " + (i+1) + "/" + MAX_POLLS +
                         " -> HTTP " + lastStatus + ". Retrying after delay…");
      TakeSometime();
      continue;
    }

    var data;
    try { data = JSON.parse(lastBody); } catch (e) {
      Context.LogMessage("Finovox /retrieve: invalid JSON on attempt " + (i+1) + ".
Retrying…");
      TakeSometime();
      continue;
    }

    var st = data.status ? ("" + data.status).toLowerCase() : "ready";
    if (st === "processing" || st === "queued" || st === "pending") {
      Context.LogMessage("Finovox /retrieve status = " + st + " (attempt " + (i+1) + ").
Waiting…");
      TakeSometime();
      continue;
    }

    return data; // Ready
  }

  throw new Error("Finovox /retrieve: gave up after " + MAX_POLLS +
                  " attempts. Last HTTP " + lastStatus + " Body: " + lastBody);
}

// --------- Field writing ----------
function WriteFields(doc, data) {
  var globalRisk = (data && data.global_risk) != null ? data.global_risk : "";
  var explanation = normaliseExplanation(data ? data.explanation : null);
  var link        = (data && data.share_link) != null ? data.share_link : "";

  safeSet(doc, "Fraud Detection/Global Risk", globalRisk);
  safeSet(doc, "Fraud Detection/Flags",       explanation);
  safeSet(doc, "Fraud Detection/Link",        link);
}

// --------- Helpers ----------
```

```
function normaliseExplanation(exp) {
  if (exp == null) return "";
  if (typeof exp === "string") return exp;

  function firstStringValue(obj) {
    for (var k in obj) if (obj.hasOwnProperty(k) && typeof obj[k] === "string" && obj[k])
return obj[k];
    return "";
  }

  function renderItem(it) {
    if (it == null) return "";
    if (typeof it === "string") return it;

    if (typeof it === "object") {
      var category = it.category || it.type || "";
      var data = it.data;
      var text = "";

      if (data != null) {
        if (typeof data === "string") {
          text = data;
        } else if (typeof data === "object") {
          text = data.en || firstStringValue(data);
        }
      }

      var prefix = category ? (category + ": ") : "";
      if (text) return prefix + text;

      try { return prefix + JSON.stringify(it); } catch (e) { return prefix + ("" + it); }
    }

    return "" + it;
  }

  if (Object.prototype.toString.call(exp) === "[object Array]") {
    var parts = [];
    for (var i = 0; i < exp.length; i++) {
      var piece = renderItem(exp[i]);
      if (piece) parts.push(piece);
    }
    return parts.join("\n"); // line breaks
  }

  return renderItem(exp);
}

function safeSet(doc, path, val) {
  try {
    var f = doc.GetField(path);
    if (f) f.Value = (val == null ? "" : ("" + val));
  } catch (e) {
    Context.LogMessage("Field not writable or not found: " + path + " (" + e + ")");
  }
}

RunFlow();
```